Fiorano
Enabling change at the speed of thought
www.fiorano.com

FioranoMQ

Handbook

AMERICA'S
Fiorano Software, Inc.
230 S. California Avenue,
Suite 103, Palo Alto,
CA 94306 USA
Tel: +1 650 326 1136
Fax: +1 646 607 5875
Toll-Free: +1 800 663 3621
Email: info@fiorano.com

EMEA
Fiorano Software Ltd.
3000 Hillswood Drive Hillswood
Business Park Chertsey Surrey
KT16 0RS UK
Tel: +44 (0) 1932 895005
Fax: +44 (0) 1932 325413
Email: info_uk@fiorano.com

APAC
Fiorano Software Pte. Ltd.
Level 42, Suntec Tower Three 8
Temasek Boulevard 038988
Singapore
Tel: +65 68292234
Fax: +65 68292235
Email: info_asiapac@fiorano.com

Contents

Chapter 1: Configuring FioranoMQ Through JMX Tools

FioranoMQ comes with full JMX support that enables management through standard JMX tools. FioranoMQ is comprised of a number of JMX enabled components running within the Fiorano container. Each component performs particular operations while displaying its attributes and enabling administration and monitoring through standard JMX tools.

The next section explains how to use the JMX based tools listed below through the RMIConnector:

- Fiorano JMX Browser
- MC4J
- JConsole

Fiorano Studio is bundled with the JMX browser, which creates a JMX connection with FioranoContainer via either RMI or JMS connectors.

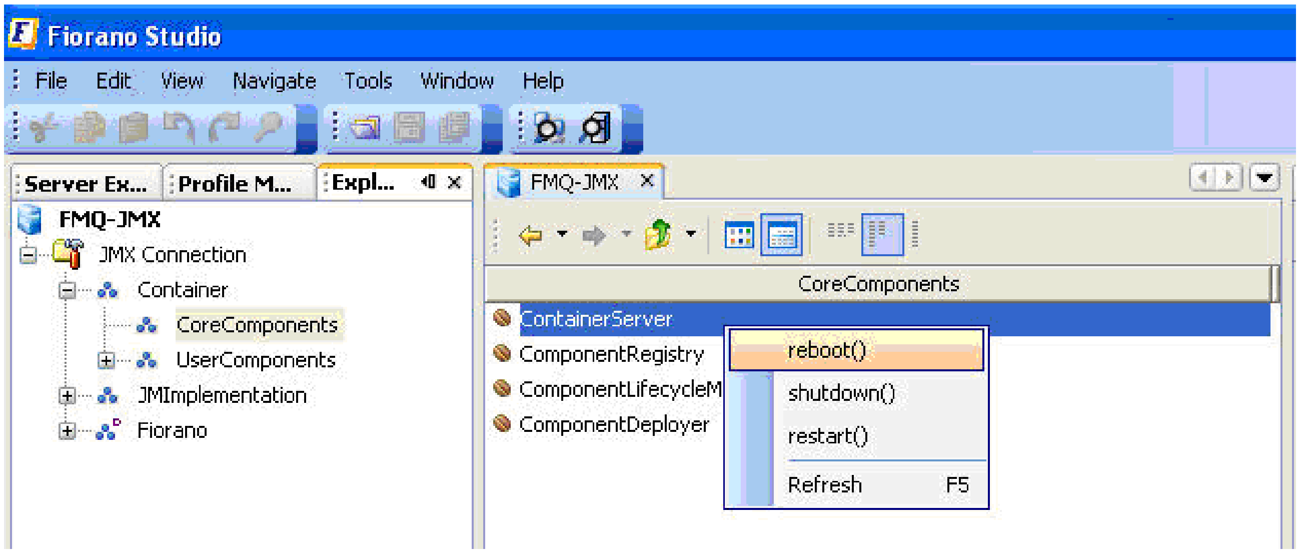
1.1 RMI Connector URL

RMIConnector connect URL – service:jmx:rmi:///jndi/rmi://<Server>:<Port>/fmq. The server can be either the machine name or the IP and the port where the RMIConnector Server is running.

1.2 Fiorano JMX Browser

Fiorano Studio supports a JSR 160 connection to the FioranoMQ Server through the RMIConnector. Please follow the instructions given below to create a JMX Connection.

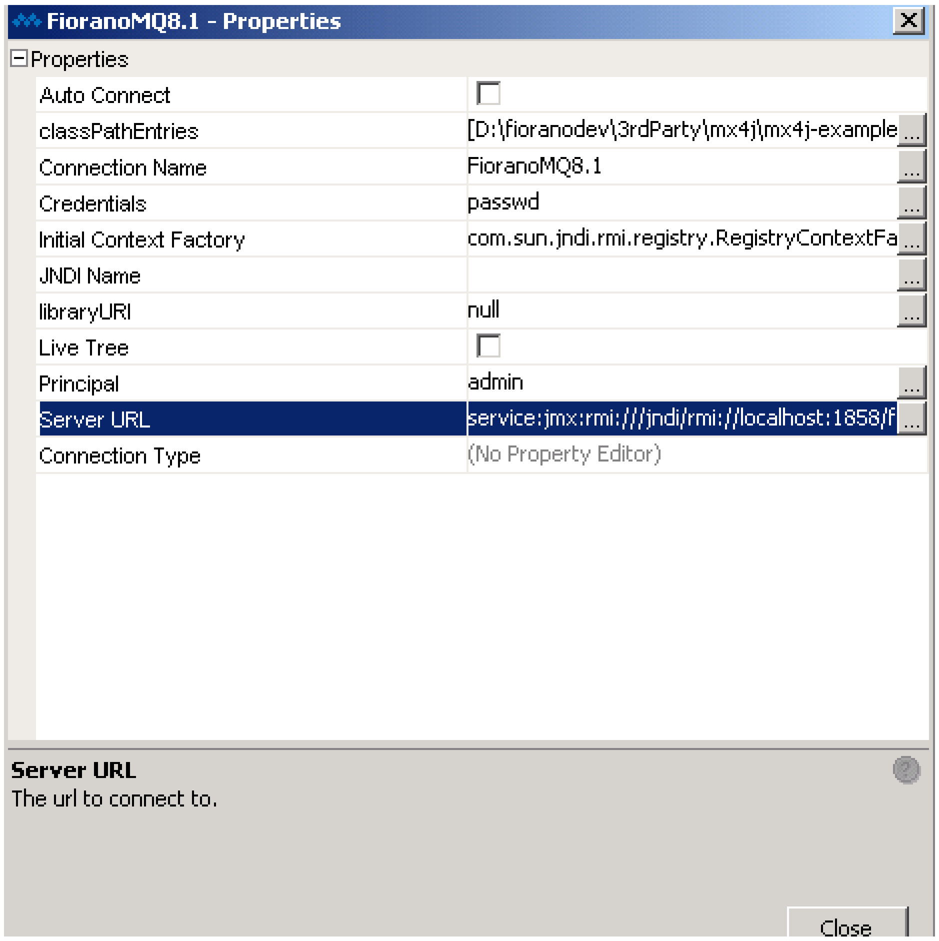
1. Launch Fiorano Studio shipped with the FioranoMQ.
2. In the **Server** explorer, right-click on the pre-existing **Fiorano JMX Connection** and select **login**. Once connected, the JMX tree is exposed with FioranoMQ MBeans as nodes.



1.3 MC4J

MC4J is an open source JMX browser used for connecting to FioranoMQ. To connect to FioranoMQ, perform the steps below:

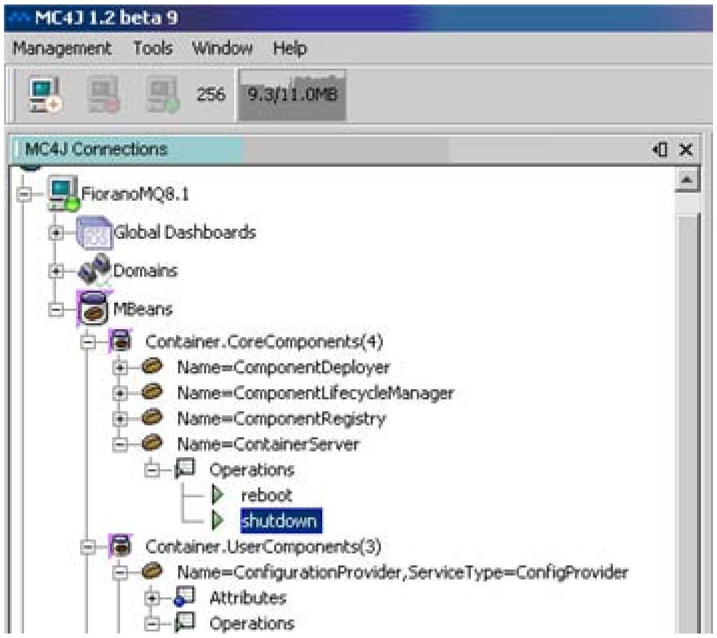
1. Install MC4J from https://sourceforge.net/projects/mc4j/ and launch the same.
2. Create a new Server connection metadata object by choosing a JSR 160 type connection.
3. You must provide the JMX Connect URL and JMX jars in the classpath. You can also provide FioranoMQ jars in the classpath as some of the exposed JMX operations can depend on them.
4. Provide the FioranoMQ Admin username and password to connect. This results in the MBean Tree view.


FioranoMQ8.1 - Properties

Properties

Auto Connect	☐
classPathEntries	[D:\fioranodev\3rdParty\mx4j\mx4j-example
Connection Name	FioranoMQ8.1
Credentials	passwd
Initial Context Factory	com.sun.jndi.rmi.registry.RegistryContextFa
JNDI Name	
libraryURI	null
Live Tree	☐
Principal	admin
Server URL	service:jmx:rmi:///jndi/rmi://localhost:1858/f
Connection Type	(No Property Editor)

Server URL
The url to connect to.

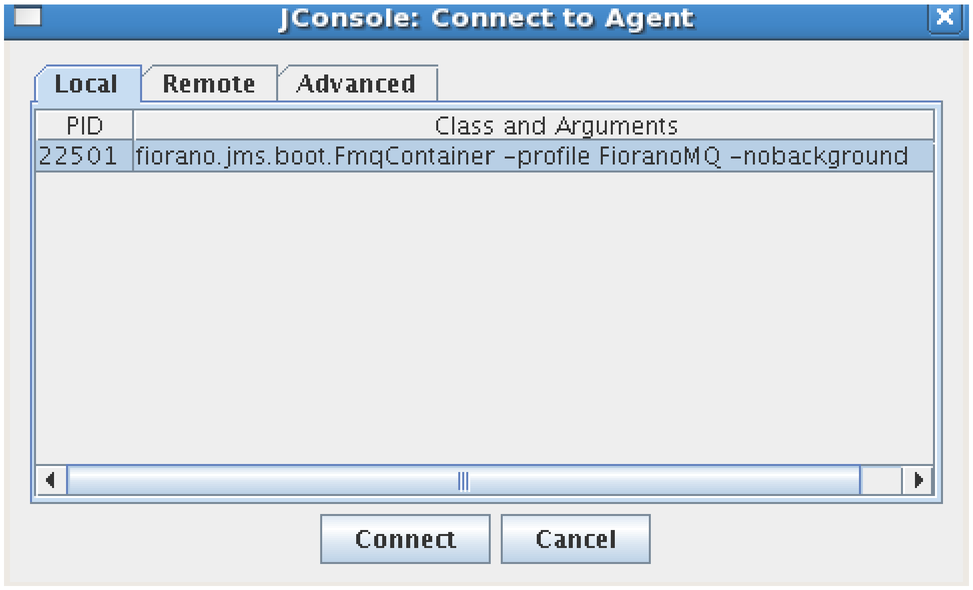


- FioranoMQ8.1
  - Global Dashboards
  - Domains
  - MBeans
    - Container.CoreComponents(4)
      - Name=ComponentDeployer
      - Name=ComponentLifecycleManager
      - Name=ComponentRegistry
      - Name=ContainerServer
        - Operations
          - reboot
          - shutdown
    - Container.UserComponents(3)
      - Name=ConfigurationProvider,ServiceType=ConfigProvider
        - Attributes
        - Operations

1.4 JConsole

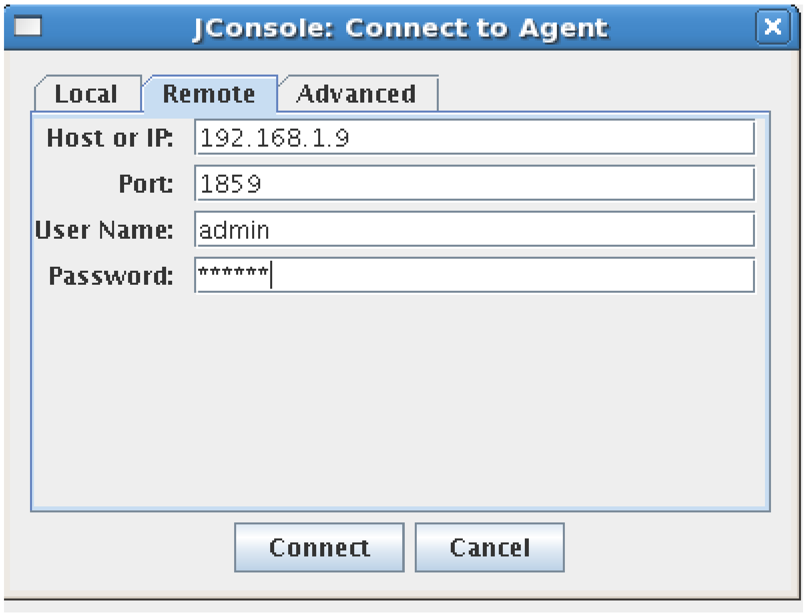
JConsole is a JMX browser from Sun Microsystems (now Oracle) and comes bundled with jdk1.5.

1. Add the following lines in the '$FIORANOHOME/fmq/bin/fmq.conf' under <java.system.props> tag.
   com.sun.management.jmxremote.port=portNum
   com.sun.management.jmxremote.authenticate=false
   com.sun.management.jmxremote.ssl=false
2. Start the FioranoMQ Server
3. Run JConsole from JDKInstallation\bin\JConsole
4. Create a server connection using one of the three ways.
   a. **Local Monitoring**: Jconsole can be used to monitor a local Java platform, that is, a JVM running on the same machine as shown below



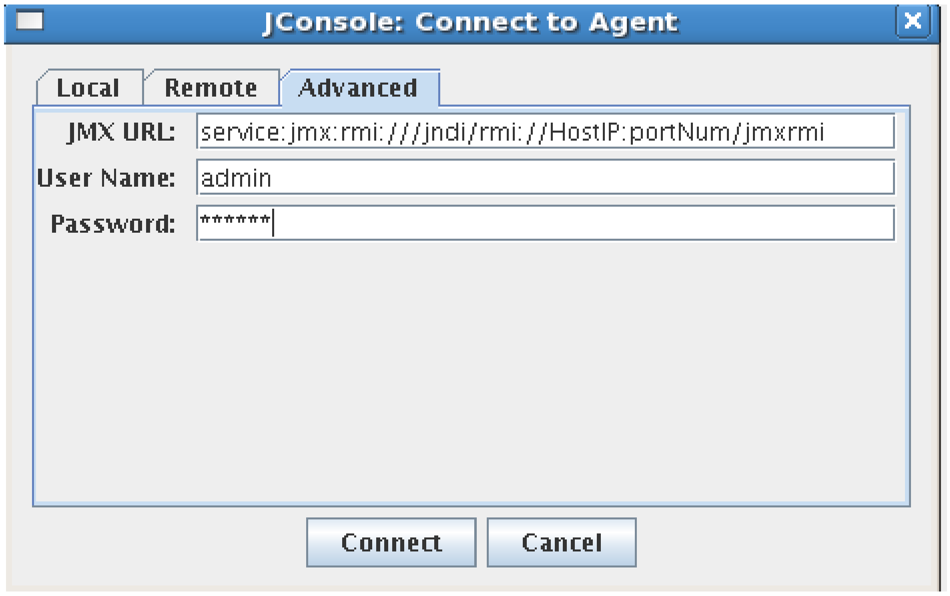
PID	Class and Arguments
22501	fiorano.jms.boot.FmqContainer -profile FioranoMQ -nobackground

b. **Remote Monitoring**: Give the Host IP Address, portNum, FioranoMQ Admin username and password, and connect as shown in the figure below:



Host or IP: 192.168.1.9
Port: 1859
User Name: admin
Password: *******

c. **Advanced Option**: Provide the JMX URL as shown where HostIP is the IPAddress of the server machine. PortNum is the port number provided in fmq.conf. Give the FioranoMQ Admin username and password, and connect.

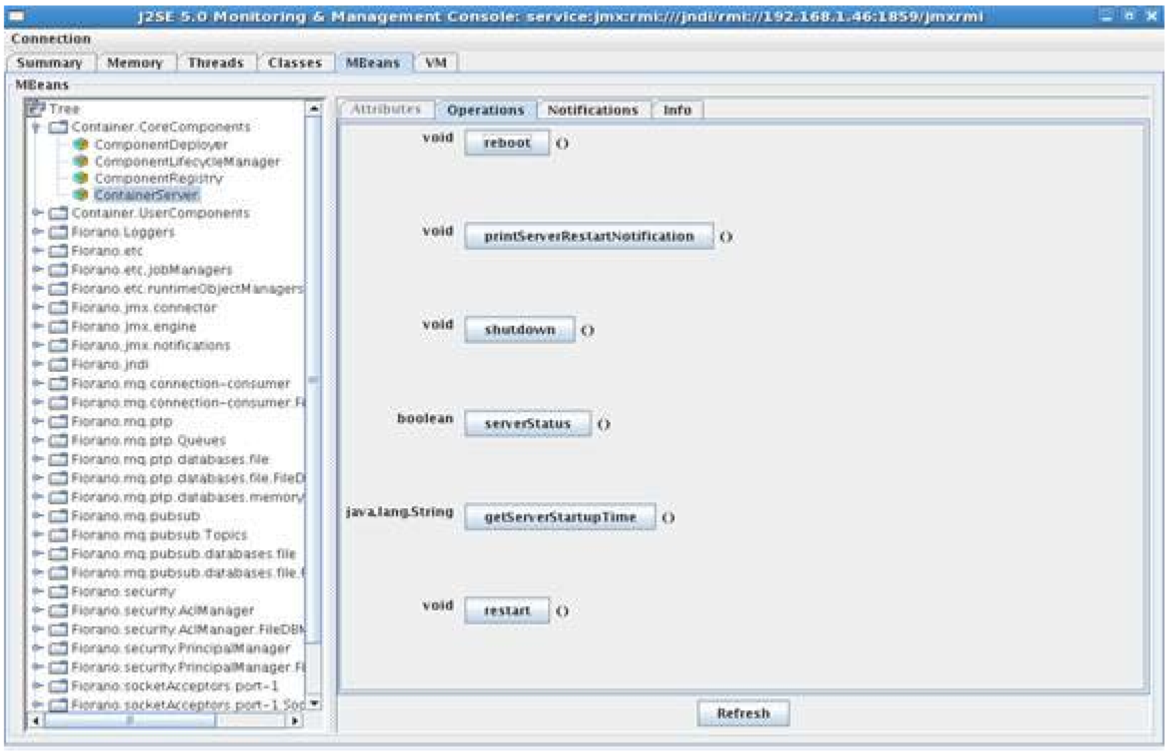

JConsole: Connect to Agent

Local | Remote | Advanced

JMX URL: service:jmx:rmi:///jndi/rmi://HostIP:portNum/jmxrmi
User Name: admin
Password: ******

Connect | Cancel

5. This results in MBeans explorer opens up as shown below


J2SE 5.0 Monitoring & Management Console: service:jmx:rmi:///jndi/rmi://192.168.1.46:1859/jmxrmi

Connection

Summary | Memory | Threads | Classes | MBeans | VM

MBeans

Tree
- Container.CoreComponents
  - ComponentDeployer
  - ComponentLifecycleManager
  - ComponentRegistry
  - ContainerServer
- Container.UserComponents
- Fiorano.Loggers
- Fiorano.etc
- Fiorano.etc.JobManagers
- Fiorano.etc.runtimeObjectManagers
- Fiorano.jmx.connector
- Fiorano.jmx.engine
- Fiorano.jmx.notifications
- Fiorano.jndi
- Fiorano.mq.connection-consumer
- Fiorano.mq.connection-consumer.F
- Fiorano.mq.ptp
- Fiorano.mq.ptp.Queues
- Fiorano.mq.ptp.databases.file
- Fiorano.mq.ptp.databases.file.FileD
- Fiorano.mq.ptp.databases.memory
- Fiorano.mq.pubsub
- Fiorano.mq.pubsub.Topics
- Fiorano.mq.pubsub.databases.file
- Fiorano.mq.pubsub.databases.file.F
- Fiorano.security
- Fiorano.security.AclManager
- Fiorano.security.AclManager.FileDBM
- Fiorano.security.PrincipalManager
- Fiorano.security.PrincipalManager.F
- Fiorano.socketAcceptors.port-1
- Fiorano.socketAcceptors.port-1.Soc

Attributes | Operations | Notifications | Info

void	reboot	()
void	printServerRestartNotification	()
void	shutdown	()
boolean	serverStatus	()
java.lang.String	getServerStartupTime	()
void	restart	()

Refresh

1.5 Java Code

Since FioranoMQ is JSR 160 compliant, the user can write simple code for working with exposed components. FioranoMQ samples for working with exposed components can be found at FIORANO_HOME\fmq\samples\JMX.

1.6 Using the FMQTerminal

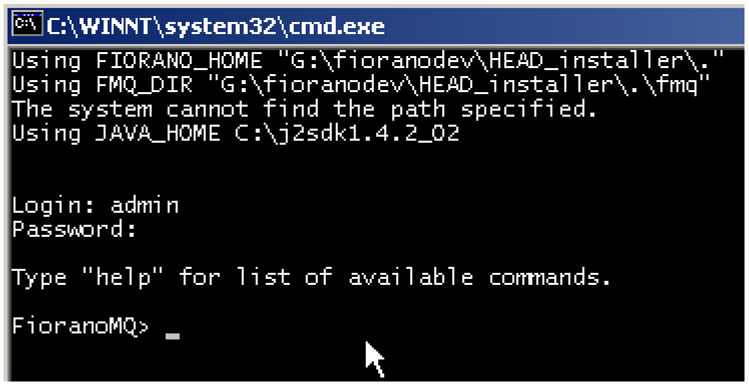
The FMQTerminal uses a third party library jline http://jline.sourceforge.net/. It is not a pure java library as it ships with some native code. The FMQTerminal works perfectly on the platforms below:

- Microsoft Windows
- RedHat Linux 9.0
- Debian Linux 3.0
- Macintosh OS X 10.3

1.6.1 How to Log in

1. Navigate to %FIORANO_HOME%\fmq\terminal\FMQConsole.bat for Windows and $FIORANO_HOME/fmq/terminal/FMQConsole.sh for UNIX systems.
2. The command without any arguments connects to the FioranoMQ Server running on localhost. A request for login and password appears. Enter login and password details.

Note: The password (typed) is not visible on the screen.



```
C:\WINNT\system32\cmd.exe
Using FIORANO_HOME "G:\fioranodev\HEAD_installer\."
Using FMQ_DIR "G:\fioranodev\HEAD_installer\.\fmq"
The system cannot find the path specified.
Using JAVA_HOME C:\j2sdk1.4.2_02


Login: admin
Password:

Type "help" for list of available commands.

FioranoMQ> _
```

The FMQTerminal works similar to normal telnet emulators like puttytel and others.

You can specify the FioranoMQ Server to connect to by passing the parameter **mars** to the executable fmq-terminal.bat (fmq-terminal.sh for Unix)

FMQConsole mars

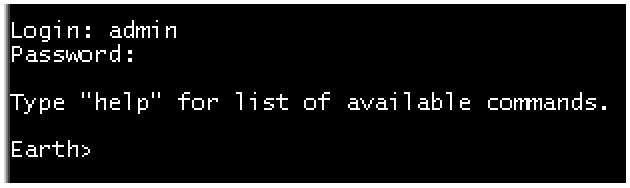
This command searches for mars.properties in the **%FIORANO_HOME%\fmq\terminal** directory. If no argument is specified the default the localhost.properties file in the **%FIORANO_HOME%\fmq\terminal** directory is used.

localhost.properties is shown below:

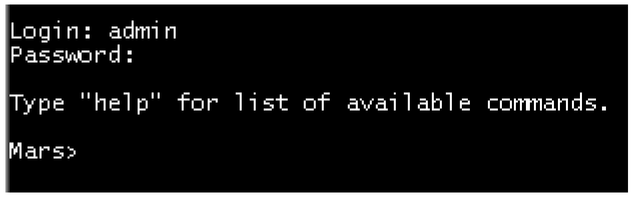
```
jmx.service.url=service:jmx:rmi://localhost/jndi/fmq

java.naming.factory.initial=com.sun.jndi.rmi.registry.RegistryContextFactory
java.naming.provider.url=rmi://localhost:1858
```

java.naming.security.principal=admin
java.naming.security.credentials=passwd

It is possible to create a properties file for each fmq server of interest and use it by passing the file name (without the extension) as an argument.



```
Login: admin
Password:

Type "help" for list of available commands.

Earth>
```



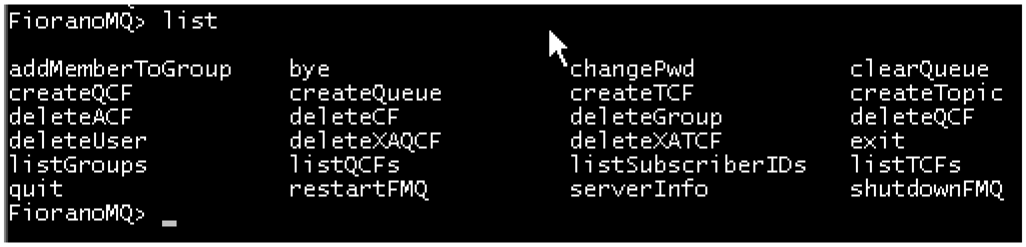
```
Login: admin
Password:

Type "help" for list of available commands.

Mars>
```

If multiple servers need to be administered from a single location, the prompt can be customized to identify the target server.

FMQConsole localhost Mars

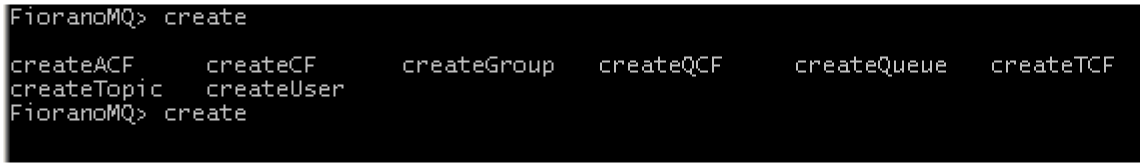
Here, the third argument specifies the prompt name.

1.6.2 Command to view a list of all Commands



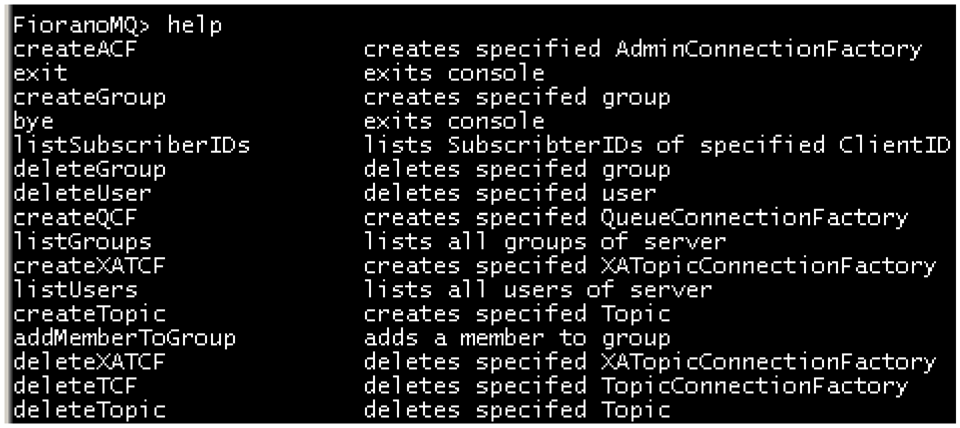
```
FioranoMQ> list

addMemberToGroup    bye                 changePwd           clearQueue
createQCF           createQueue         createTCF           createTopic
deleteACF           deleteCF            deleteGroup         deleteQCF
deleteUser          deleteXAQCF         deleteXATCF         exit
listGroups          listQCFs            listSubscriberIDs   listTCFs
quit                restartFMQ          serverInfo          shutdownFMQ
FioranoMQ> _
```

Type a few characters and press the <tab> key for auto completion.



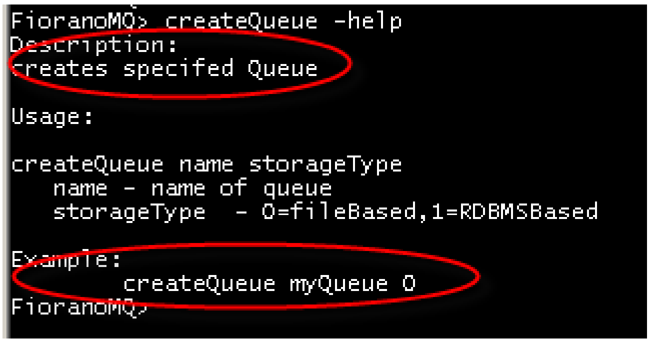
```
FioranoMQ> create

createACF      createCF       createGroup    createQCF      createQueue    createTCF
createTopic    createUser
FioranoMQ> create
```

1.6.3 Help Command to Print Command and their Help



```
FioranoMQ> help
createACF                creates specified AdminConnectionFactory
exit                     exits console
createGroup              creates specifed group
bye                      exits console
listSubscriberIDs        lists SubscribterIDs of specified ClientID
deleteGroup              deletes specifed group
deleteUser               deletes specifed user
createQCF                creates specifed QueueConnectionFactory
listGroups               lists all groups of server
createXATCF              creates specified XATopicConnectionFactory
listUsers                lists all users of server
createTopic              creates specifed Topic
addMemberToGroup         adds a member to group
deleteXATCF              deletes specifed XATopicConnectionFactory
deleteTCF                deletes specifed TopicConnectionFactory
deleteTopic              deletes specifed Topic
```

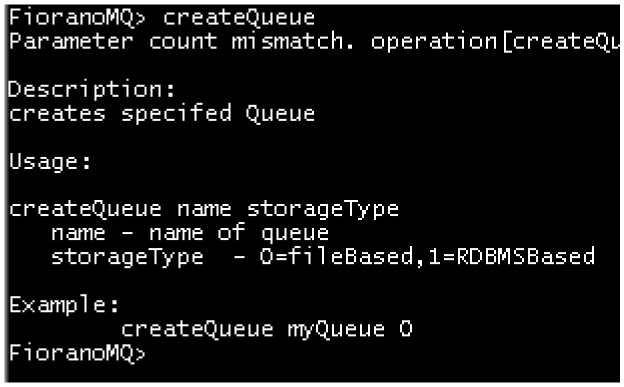
1.6.4 Description and Example for Command



```
FioranoMQ> createQueue -help
Description:
creates specifed Queue

Usage:

createQueue name storageType
   name - name of queue
   storageType  - 0=fileBased,1=RDBMSBased

Example:
        createQueue myQueue 0
FioranoMQ>
```

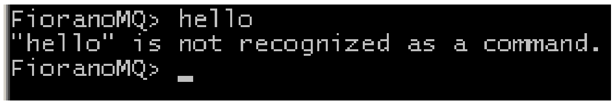
The examples above show how arguments can be passed.

1.6.5 Print Usage on Parameter Mismatch



```
FioranoMQ> createQueue
Parameter count mismatch. operation[createQu

Description:
creates specifed Queue

Usage:

createQueue name storageType
   name - name of queue
   storageType  - 0=fileBased,1=RDBMSBased

Example:
        createQueue myQueue 0
FioranoMQ>
```

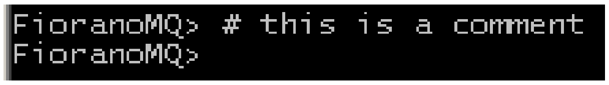
Invalid commands are displayed as ‘not recognized’:



```
FioranoMQ> hello
"hello" is not recognized as a command.
FioranoMQ> _
```

1.6.6 Comment Command

All lines starting with # (a Hash or Pound sign) are treated as a comment and do not perform an action.



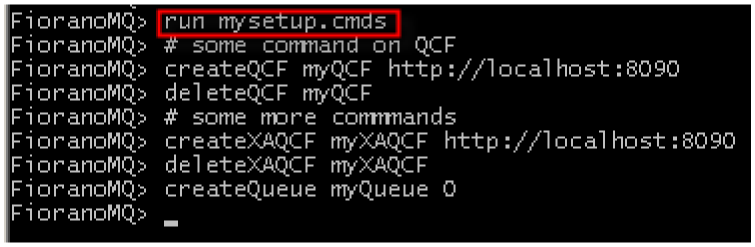
```
FioranoMQ> # this is a comment
FioranoMQ>
```

1.6.7 Run List of Commands from File

If there is a file named mysetup.cmds with the following contents:

```
 # Some commands on QCF
createQCF myQCF http://localhost:8090
deleteQCF myQCF


# Commands continued
createXAQCF myXAQCF http://localhost:8090
deleteXAQCF myXAQCF
createQueue myQueue 0
```

Then, it is possible to run the following commands on this file:



```
FioranoMQ> run mysetup.cmds
FioranoMQ> # some command on QCF
FioranoMQ> createQCF myQCF http://localhost:8090
FioranoMQ> deleteQCF myQCF
FioranoMQ> # some more commmands
FioranoMQ> createXAQCF myXAQCF http://localhost:8090
FioranoMQ> deleteXAQCF myXAQCF
FioranoMQ> createQueue myQueue 0
FioranoMQ> _
```

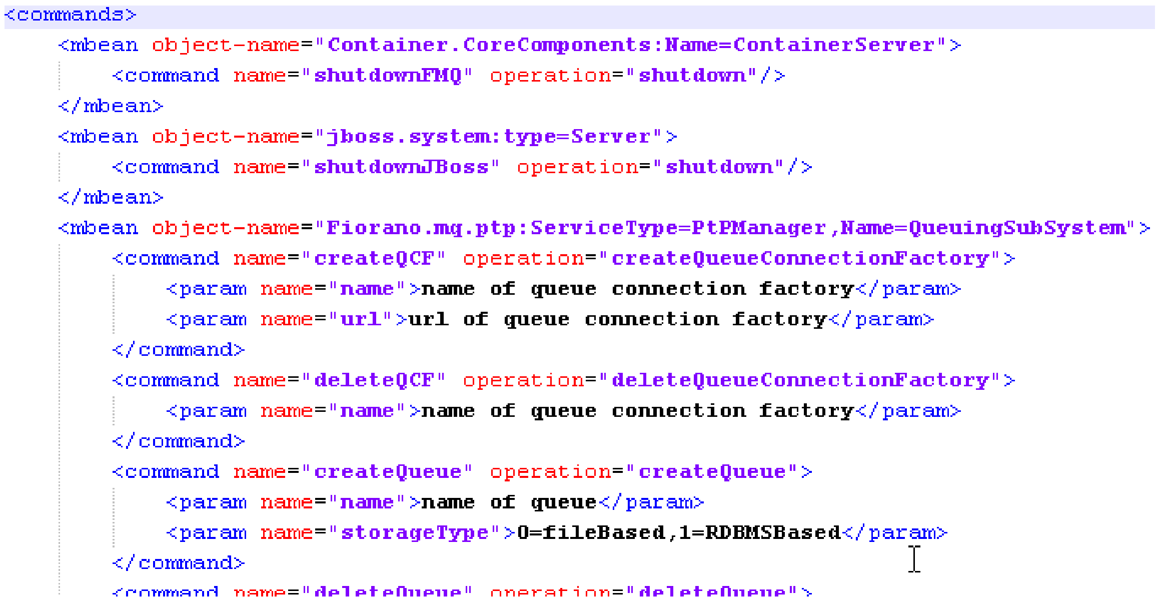
In the above example, -quiet can be specified as a second argument if commands are not to be echoed.

1.6.8 Adding more Commands to a Terminal

All the available commands can be viewed in the **commands.xml** file:

%FIORANO_HOME%\fmq\terminal\commands.xml

A snippet of **commands.xml** is shown below:



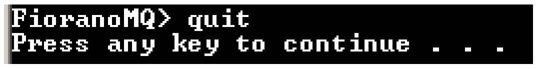
```
<commands>
    <mbean object-name="Container.CoreComponents:Name=ContainerServer">
        <command name="shutdownFMQ" operation="shutdown"/>
    </mbean>
    <mbean object-name="jboss.system:type=Server">
        <command name="shutdownJBoss" operation="shutdown"/>
    </mbean>
    <mbean object-name="Fiorano.mq.ptp:ServiceType=PtPManager,Name=QueuingSubSystem">
        <command name="createQCF" operation="createQueueConnectionFactory">
            <param name="name">name of queue connection factory</param>
            <param name="url">url of queue connection factory</param>
        </command>
        <command name="deleteQCF" operation="deleteQueueConnectionFactory">
            <param name="name">name of queue connection factory</param>
        </command>
        <command name="createQueue" operation="createQueue">
            <param name="name">name of queue</param>
            <param name="storageType">0=fileBased,1=RDBMSBased</param>
        </command>
        <command name="deleteQueue" operation="deleteQueue">
```

The Mbean operations are found in the studio and can be added to the **commands.xml** file. This function ensures not being limited to the built-in commands that ship with the installer.

1.6.9 Exiting the Terminal

To exit the FMQTerminal, following commands can be used:

- exit, quit, bye, or pressing ctrl+D



```
FioranoMQ> quit
Press any key to continue . . .
```

This returns you to the standard prompt command.

Chapter 2: Using Scripts

This chapter explains the new scripts introduced in FioranoMQ that manage and configure the FioranoMQ Server. These scripts replace previous scripts, with the following benefits:

- A common configuration file is now used to specify properties such as classpath, memory settings, and system properties in OS, in independent manner.
- OS specific files like .bat and .sh do not need to be modified given a change in configuration. Changes are now specified in configuration file.
- The Configuration file can be reused across different servers and tools.
- No prior knowledge of OS specific details required to configure the server/tools properties, since the configuration file supports a simple and easy to use syntax.

2.1 Configuration Files

Each script in FioranoMQ is associated with a specific configuration file (.conf) in the same directory as the script file. This has the same name as the script file followed with .conf extension. This configuration file includes configuration properties of server/tools, as listed below:

Config Property/Block	**Usage**
<java.classpath>	Specifies any additional jar files required in classpath in separate lines at the end of this block.
<java.endorsed.dirs>	Specifies the jars to be considered other than the default jars in separate lines at the end of this block.
<java.ext.dirs>	Specifies the external jar files to be loaded along with default system jars in separate lines as the end of the block.
<java.library.path>	Specifies the folders containing dll/so files to be loaded in separate lines at the end of the block.
<java.system.props>	Specifies any additional system properties, in separate lines at the end of the block.
<jvm.args>	Specifies arguments to the JVM such as memory settings, debug info etc. in separate lines at the end of the block.

Note the following points about the configuration file:

- A line starting with '#' is treated as a comment.
- The Conf file can have empty lines. These empty lines are ignored by the launcher.
- Environment variables can be used in the conf file. (Using environment variables makes conf file OS dependent).
- Wild-cards are not supported. Example: lib/*.jar is disallowed.

2.2 Reference Matrix - UNIX

The table below summarizes all the scripts/configurations that are changed in FioranoMQ for **UNIX**:

Functionality	**Old Script (Before FioranoMQ 2008)**	**New Script (From FioranoMQ 2008)**
	FioranoMQ Server	
Memory settings of FioranoMQ Server	/fiorano_vars.sh	/fmq/bin/fmq.conf
External jar files	/fmq/bin/fmq.sh	/fmq/bin/fmq.conf
Startup	/fmq/bin/fmq.sh	/fmq/bin/fmq.sh
Shutdown	/fmq/bin/shutdownFMQ.sh	/fmq/bin/shutdown-fmq.sh
	Tools and Miscellaneous Scripts	
Fiorano Studio	/Studio/bin/Studio.sh	/Studio/bin/Studio.sh
Compile Client	/fmq/bin/compClient.sh	/fmq/bin/compile-client.sh
External Jar files for Client	/fmq/bin/compClient.sh	/fmq/bin/compile-client.conf
Run Client	/fmq/bin/runClient.sh	/fmq/bin/run-client.sh
External Jar files for Client	/fmq/bin/runClient.sh	/fmq/bin/run-client.sh
Memory settings for Client	/fiorano_vars.sh	/fmq/bin/run-client.conf
Run Standalone Bridge	/fmq/bin/runStandaloneBridge.sh	/fmq/bin/fmq.sh –profile StandAloneBridge
Run Standalone Repeater	/fmq/bin/runStandaloneRepeater.sh	/fmq/bin/fmq.sh –profile StandAloneRepeater
	Database related scripts	
Create Database	/fmq/bin/createDB.sh	/fmq/bin/create-database.sh
External Jar files to Create Database	/fmq/bin/createDB.sh	/fmq/bin/create-database.conf
DB Recovery Tool	/fmq/bin/runDBRecoveryTool.sh	/fmq/bin/recover-database.sh
External Jar files for Database Recovery	/fmq/bin/ runDBRecoveryTool.sh	/fmq/bin/recover-database.conf

2.3 Common Scripts Usage - UNIX

Scripts present under <fiorano_installation_dir>

fiorano_vars.sh

Used to set Environment Variable which are used throughout the Fiorano server scripts. It is also used to set the Endorsed libraries.

setScriptPermissions.sh

Script used to set executable permissions for all the scripts present in the FioranoMQ Installation.

Scripts present under <fiorano_installation_dir>/fmq/bin

- **backupDB.sh:** This script is used to backup the database of the profile specified by the user. This script moves the existing database from the profile directory to the backup database directory.

  Usage: backupDB.sh <profileName> <FMQ DB path> <DB Backup Path>

  Example: backupDB.sh FioranoMQ/profiles/FioranoMQ/run/profiles/FioranoMQ_backupDB

  This moves the existing database from the FioranoMQ/run directory to a specified directory. By default, a backup of the FioranoMQ database is taken and a new directory inside the profiles/FioranoMQ directory is created using the current date of the system.

- **ClearDB.sh:** This script is used to clear the existing database of the FioranoMQ Server.

  Usage: ClearDB.sh [– profile <profileName>] [-DBDir <FMQ DB path>] [-profilesDir <Profiles directory>]. By default, the database of FioranoMQ profile is cleared upon this command.

  - **-profile <profilename>:** Profilename from profiles directory to the profile to be deployed. Defaults to FioranoMQ.
  - **–DBDir <FMQ DB path>:** Database directory for configured profile. Defaults to <fiorano_installation_dir>\fmq\profiles\<profile>\run"
  - **-profilesDir <Profiles directory>:** Directory where profiles are present. Defaults to <fiorano_installation_dir>\fmq\profiles

  Warning: Clearing the database removes all user created information, including all Topics/Queues/ConnectionFactories created, as well as User permissions and Groups. If the user wants to clear the database only partially, the option to clear Admin Storage, Data Storage, and ACL Storage can be chosen instead.

- **compile-client.sh:** This script is used to compile Java samples which are included within the installer. External jars can be added to classpath in <fiorano_installation_dir>/fmq/bin/complie-client.conf

  Usage: compile-client.sh <sampleName>.java

- **create-database.sh:** This script is used to create a database. There are two ways by which databases can be created.

a) **Using FioranoMQ Configuration:**

The database section of the FioranoMQ profile has to be configured. Refer to the chapter Configuring Message Store for information on configuring the database section*.*

Usage: create-database.sh -fmq.profile <profileName> -dataBaseType <dataBaseType>

Example: The following command creates a database using message store configuration for MSSQL in FioranoMQ_SQL profile:

create-database.sh -fmq.profile FioranoMQ_SQL -dataBaseType mssql

b) From command line arguments

Database configuration details are provided as options in the command line.

Usage: Usage: create-database -dataBaseType <DataBase> -driver <Driver> - url <URL> -userName <UserName> -password <Password> -dataTypesFileName <cfgFileName>

Please refer to the chapter on Configuring Message Store

Example: The following command creates a database using message store configuration for MSSQL:

create-database.sh -dataBaseType **mssql** -driver **com.microsoft.jdbc.sqlserver.SQLServerDriver** -url **jdbc:microsoft:sqlserver://server:1433;SelectMethod=Cursor** -userName **admin** -password **passwd** -dataTypesFileName **jdbc_mssqls.cfg**

- **fmq.sh:** Launches the FioranoMQ server. By default, FioranoMQ profile is launched if no other profile is specified.

  Usage: fmq.sh [-profilesDir <dir>] [-profile <path>] [-dbPath <dir>] [-configPath <dir>] [-saveConfigs <boolean>]

  - **-profile <path>:** Relative path from profiles directory to the profile to be deployed. Defaults to FioranoMQ.
  - **-profilesDir <dir>**: Directory where profiles are present. Defaults to <fiorano_installation_dir>/fmq/profiles
  - **-configPath <dir>:** Conf directory path for the configured profile. Defaults to <fiorano_installation_dir>/fmq/profiles/<profile>/conf
  - **-dbPath <dir>:** Database path for configured profile. Defaults to <fiorano_installation_dir>/fmq/profiles/<profile>/run"
  - **-saveConfigs <boolean>:** Persists server configurations on shutdown. Defaults to false.
  - **–nobackground:** Option to not start the server in the back ground.
  - **–help:** Displays all available options.

Example: Following command starts FioranoMQ HA primary server.
fmq.sh –profile FioranoMQ_HA_rpl/HAPrimary

- **recover-database.sh:** This script is used to recover the database in the event that the FioranoMQ proprietary File based database is corrupt.

  This launches the script and, by default, recovers the PTP database. For more information on the usage of the script refer to Chapter 24: Fiorano DB Recovery Tool.

  Usage: recover-database.sh [–propertiesFile <path>] [-fmq.profile <path>] [-h]

  - **–propertiesFile:** Configuration file. Default configuration file is located at <fiorano_installation_dir>/fmq/profiles/recovery.properties
  - **-fmq.profile <path>:** Relative path from the profiles directory to profile to be deployed. Defaults to FioranoMQ.
  - **-h:** display help

- **run-client.sh:** This script is used to run Java samples which are included in the FioranoMQ installation. External jars can be added to classpath in fiorano_installation_dir>/fmq/bin/run-client.conf

  Usage: run-client.sh <sampleName>

- **build-all-clients.sh**

  This script is used to compile all the Java samples which are included within the installer. The script makes use of compile-client.sh internally to compile each of the samples. External jars can be added to classpath in <fiorano_installation_dir>/fmq/bin/complie-client.conf

  Usage: build-all-clients.sh

- **migrate.sh and convert-database.sh**

  The migrate.sh script file is used for migrating the FioranoMQ2007 database to a FioranoMQ 9.0 or higher versions, which uses the FioranoMQ9 compatible format. This script uses convert-database.sh for converting the database table files and data files to the FioranoMQ9 compatible format. It is recommended that the readme.txt file be referred to for instructions on using this utility.

- **startCluster.sh**

  The startCluster.sh script file is used for running the FioranoMQ Management server using FioranoMQ_Clustering profile. For more information on the FioranoMQ Management server and its functions refer to Chapter 32: Fiorano Directory Services.

- **routeUtility.sh**

  routeUtility.sh script file is used for creating/removing multiple routes between destinations (Queues/Topics) based on the configuration file specified while running the 'route' utility. The configuration file (routes.xml) contains properties of each route to be created.

  A sample route configuration file can be found at $FIORANO_HOME/fmq/Utilities/RouteUtility/conf folder.

  Usage: $FIORANO_HOME/fmq/bin/routeUtility.sh [-operation <operation> -configFile <routeConfigsFile>]

  where:

configFile: Includes the XML file that contains the configuration of the routes. If no configuration file is specified when running the route Utility then 'routes.xml' will be taken as the default configuration file.

Operation: Operation that needs to be executed using the route utility.

Valid Values: createRoutes OR removeRoutes. If no operation is specified, then this will take 'createRoutes' as its operation.

- **shutdown-fmq.sh**

  This script is used to shutdown the FioranoMQ server This script internally creates a JMXConnection based on arguments passed. If none of the arguments are passed, then the default parameters are used. If the arguments passed are invalid, then an exception is shown on the console.

  Usage: ./shutdown-fmq.sh [-options] (For Linux)

Where options include:

Parameter Name	**Description**	**Legal Values**
connectorType	Type of connector that needs to be created for making a jmx connection with the FioranoMQ server.	RMI (Default Value), JMS
host	URL of the machine where the server is running. If the server is running on a machine on which a user executes a shutdown script, the host need not be specified.	localhost (Default Value), IPAddress
port	The port on the FioranoMQ server that will accept JMS connections. This value should be specified only if the connector 'type' is JMS.	1856 (Default Value)
user	Name of the user trying to shutdown the FioranoMQ server. This user should have permission to create a JMX connection.	admin (Default Value), anonymous, ayrton
passwd	Password of the user trying to shutdown the FioranoMQ Server.	passwd(Default Value for admin), anonymous (for user anonymous), senna (for user ayrton)
transportProtocol	Transport protocol to be used for making a connection with the FioranoMQ server. This value should be specified only if the connector 'type' is JMS.	TCP (Default Value), HTTP, LPC

Parameter Name	Description	Legal Values
securityProtocol	Security protocols that need to be used for making a connection with the FioranoMQ server. By default there is no security protocol. This value should be specified only if the connector 'type' is JMS.	null (Default Value), SUN_SSL.
securityManager	Security Manager which to be used for making a connection with the FioranoMQ server. By default there is no security Manager. This value should be specified only if the connector 'type' is JMS.	Null (Default Value), DefaultJSSESecurityManager
rmiPort	RMIConnector port on which the FioranoMQ server JMX connection can be created. This value should be specified only if the connector 'type' is RMI.	1858(Default value), 2059 is default value for Repeater

Example:

The commands below stop the FioranoMQ server running on a FMQLinux1 machine on default port 1856 with user as admin and password as passwd.

./shutdown-fmq.sh -host FMQLinux1 -port 1856 -user admin -passwd passwd -transportProtocol TCP -securityProtocol null -securityManager null.

No securityManager or securityProtocol is required in this case and both are specified as null. This script creates an RMIBasedJMXConnector since the connector type is not specified.

Note: If a user has configured the FioranoMQ server to accept JMSBasedJMXConnector connections, shutting down the FioranoMQ server is possible only by using this same connector. Otherwise, by default, the RMIBasedJMXConnector will be used.

Scripts present under <fiorano_installation_dir>/fmq/clients/c/crosscomp/scripts

- **cclientbuild.sh:** This script is used to compile the C Cross Comp samples.
- **Usage:** cclientbuild.sh ../samples/<sample_type>/<sampleName>.c

Scripts present under <fiorano_installation_dir>/fmq/clients/c/native/scripts

- **cclientbuild.sh:** This script is used to compile the C Native samples.
- **Usage:** cclientbuild.sh ../samples/<sample_type>/<sampleName>.c

Scripts present under <fiorano_installation_dir>/fmq/clients/cpp/jni/scripts

- **cppclientbuild.sh:** This script is used to compile the Cpp JNI samples using fmq-jni-cpprtl.lib.
- **Usage:** cppclientbuild.sh ../samples/<sample_type>/<sampleName>.cpp

Scripts present under <fiorano_installation_dir>/fmq/terminal

- **fmq-terminal.sh:** This script is used to perform most of the admin operations that can be performed through the Studio.
- **Usage:** fmq-terminal.sh. Queries the username and password needed to log onto the server.

Note:

- help command lists all the commands/operations supported by the terminal
- <command/operation> -help displays usage of the command or its operation

Below is a list of operations supported by fmq-terminal:

Operation	**Description**
addMemberToGroup	adds a member to a Group
Bye	exits the console
changePwd	changes password for the specified User
clearQueue	clears messages in the Queue specified
createACF	creates the specified AdminConnectionFactory
createCommonCF	creates the specified CommonConnectionFactory
createGroup	creates the Group specified
createQCF	creates the specified QueueConnectionFactory
createQueue	creates the Queue specified
createTCF	creates the specified TopicConnectionFactory
createTopic	creates the Topic specified
createUser	creates a new User
createXAQCF	creates the specified XAQueueConnectionFactory
createXATCF	creates the specified XATopicConnectionFactory
deleteACF	deletes the specified AdminConnectionFactory
deleteCommonCF	deletes the specified CommonConnectionFactory
deleteGroup	deletes the Group specified
deleteQCF	deletes the specified QueueConnectionFactory
deleteQueue	deletes the Queue specified
deleteTCF	deletes the specified TopicConnectionFactory
deleteTopic	deletes the Topic specified
deleteUser	deletes a specified user
deleteXAQCF	deletes the specified XAQueueConnectionFactory
deleteXATCF	deletes the specified XATopicConnectionFactory
exit	Exits the console

Operation	**Description**
help	prints the ‘help’ topics/instructions for all available commands
list	lists all available commands
listACFs	lists names of all available Admin Connection Factories
listClientIDs	lists names of all Client IDs
listCommonCFs	lists names of all available Common Connection Factories
listDomains	list all domains
listGroups	lists all server groups
listQCFs	lists names of all available QueueConnectionFactories
listQueues	lists names of all available Queues
listSubscriberIDs	lists SubscribterIDs of specified ClientIDs
listTCFs	lists names of all available TopicConnectionFactories
listTopics	lists names of all available Topics
listUsers	lists all Users of the server
listXAQCFs	lists names of all available XAQueueConnectionFactories
listXATCFs	lists names of all available XATopicConnectionFactories
mbeanCount	prints mbeans count
quit	exits the console
restartFMQ	restarts FioranoMQ server running on the Fiorano Container
run	runs the commands in the specified file
serverInfo	prints the server information
shutdownFMQ	Shuts down the FioranoMQ server running in the relevant Fiorano Container
shutdownJBoss	Shuts down the FioranoMQ Server running in the relevant JBoss Container

Scripts present under <fiorano_installation_dir>/framework/tools/LicenseManager/bin

runLM.sh: This script is used for running the license manager. For more information on the license manager refer to *SOA Platform License Manager Guide.pdf* in the <fiorano_installation_dir>/framework/tools/LicenseManager/doc directory.

Scripts present under <fiorano_installation_dir>/Studio/bin

Reset.sh: This script is used to reset all cached data from <user_dir> that defaults to /<Installation_Dir> /runtimedata/studio/<build_no>, and delete the <user_dir>.

Usage: Reset.sh

Studio.sh: This script is used to launch the FioranoMQ Studio.

Usage: Studio.sh --jdkhome <jdk_home>

The <jdk_home> should point to the JDK_HOME since it is required for launching the Fiorano Studio.

2.4 Reference Matrix - Windows

The table below summarizes all the FioranoMQ scripts/configurations modified on **Windows**:

Functionality	**Old Script (Pre FioranoMQ 2008)**	**New Script (FioranoMQ 2008 onwards)**
	FioranoMQ Server	
Memory settings of FioranoMQ Server	/fiorano_vars.bat	/fmq/bin/fmq.conf
External jar files	/fmq/bin/runContainer.bat	/fmq/bin/fmq.conf
Startup	/fmq/bin/runContainer.bat	/fmq/bin/fmq.bat
Shutdown	/fmq/bin/shutdownFMQ.bat	/fmq/bin/shutdown-fmq.bat
	FioranoMQ Server as Windows Service	
Install	/fmq/bin/ntservice/bin/install-nt.bat	/fmq/bin/service/install-fmq.service.bat
Uninstall	/fmq/bin/ntservice/bin/uninstall-nt.bat	/fmq/bin/service/uninstall-fmq.service.bat
Install a profile	/fmq/bin/ntservice/bin/install-nt.bat ../conf/<profile_name>.conf	/fmq/bin/service/install-fmq.service.bat –profile <profile_name>
Uninstall a profile	/fmq/bin/ntservice/bin/uninstall-nt.bat ../conf/<profile_name>.conf	/fmq/bin/service/uninstall-fmq.service.bat –profile <profile_name>
Default log location	/fmq/bin/ntservice/logs	/fmq/<profiles_dir>/<profile_name>/service
	Tools and Miscellaneous Scripts	
Fiorano Studio	/Studio/bin/Studio.exe	/Studio/bin/Studio.exe
Fiorano Console	/fmq/bin/fmq-console.bat	/fmq/bin/fmq-console.bat
Compile Client	/fmq/bin/compClient.bat	/fmq/bin/compile-client.bat
External Jar files for Client	/fmq/bin/compClient.bat	/fmq/bin/compile-client.conf
Run Client	/fmq/bin/runClient.bat	/fmq/bin/run-client.bat
External Jar files for	/fmq/bin/runClient.bat	/fmq/bin/run-client.conf

Functionality	**Old Script (Pre FioranoMQ 2008)**	**New Script (FioranoMQ 2008 onwards)**
Client		
Memory settings for Client	/fiorano_vars.bat	/fmq/bin/run-client.conf
Run Standalone Bridge	/fmq/bin/runStandaloneBridge.bat	/fmq/bin/fmq.bat –profile StandAloneBridge
Run Standalone Repeater	/fmq/bin/runStandaloneRepeater.bat	/fmq/bin/fmq.bat –profile StandAloneRepeater
Database related scripts		
Create Database	/fmq/bin/createDB.bat	/fmq/bin/create-database.bat
External Jar files used for creating Databases	/fmq/bin/createDB.bat	/fmq/bin/create-database.conf
DB Recovery Tool	/fmq/bin/runDBRecoveryTool.bat	/fmq/bin/recover-database.bat
External Jar files used for Database Recovery	/fmq/bin/ runDBRecoveryTool.bat	/fmq/bin/recover-database.conf

2.5 Common Scripts Usage - Windows

Scripts present under <fiorano_installation_dir>

1. **fiorano_vars.bat:** Used to set the Environment Variable which is used throughout Fiorano server scripts. It is also used to set Endorsed libraries.
2. **uninstall.bat:** Script used to uninstall the FioranoMQ installation.

Scripts present under <fiorano_installation_dir>\fmq\bin

1. **backupDB.bat:** This script is used to backup the database of the profile specified by the User. This script moves the existing database from the profile directory to the backup database directory.

   Usage: backupDB.bat <profileName> <FMQ DB path> <DB Backup Path>

   Example: backupDB.bat FioranoMQ\profiles\FioranoMQ\run\profiles\FioranoMQ_backupDB

   This moves the existing database from the FioranoMQ\run directory to the directory specified by the User. By default, a backup of the FioranoMQ database is taken and a new directory created inside profiles\FioranoMQ directory using the current date of the system.

2. **ClearDB.bat:** This script is used to clear the existing database in the FioranoMQ server.

Usage: ClearDB.bat [– profile <profileName>] [-DBDir <FMQ DB path>] [-profilesDir <Profiles directory>]. By default, this clears the FioranoMQ profile database.

- **-profile <profilename>:** Profilename from profiles directory to the profile to be deployed. Defaults to FioranoMQ.
- **–DBDir <FMQ DB path>:** Database directory of the profile configured. Defaults to <fiorano_installation_dir>\fmq\profiles\<profile>\run"
- **-profilesDir <Profiles directory>:** Directory where profiles are present. Defaults to <fiorano_installation_dir>\fmq\profiles

Warning: Clearing the database removes all user created information, including all Topics/Queues/ConnectionFactories created, as well as User permissions and Groups. If the user wants to clear the database only partially, the option to clear Admin Storage, Data Storage and ACL Storage can be chosen instead the whole database.

Example:

3. **compile-client.bat:** This script is used to compile Java samples included in the installer.

   Usage: compile-client.bat <sampleName>.java

4. **create-database.bat:** This script is used to create a database. There are two ways by which databases can be created.

   a) **Using FioranoMQ Configuration:**

   The database section of the FioranoMQ profile has to be configured. Refer to the Chapter 4: Configuring Message Store for information on configuring the database section.

   Usage: create-database.bat -fmq.profile <profileName> -dataBaseType <dataBaseType>

   Example: The following command creates a database using message store configuration for MSSQL in the FioranoMQ_SQL profile:

   create-database.bat -fmq.profile FioranoMQ_SQL -dataBaseType mssql

   b) From command line arguments

   Database configuration details are provided as an option on the command line.

   Usage: Usage: create-database -dataBaseType <DataBase> -driver <Driver> - url <URL> -userName <UserName> -password <Password> -dataTypesFileName <cfgFileName>

   Please refer to the Chapter 4: Configuring Message Store

   Example: The following command creates a database using message store configuration for MSSQL.

create-database.bat -dataBaseType **mssql** -driver **com.microsoft.jdbc.sqlserver.SQLServerDriver** -url **jdbc:microsoft:sqlserver:\\server:1433;SelectMethod=Cursor** -userName **admin** -password **passwd** -dataTypesFileName **jdbc_mssqls.cfg**

5. **fmq.bat:** Launches the FioranoMQ server. By default, the FioranoMQ profile is launched if no other profile is specified.

   Usage: fmq.bat [-profilesDir <dir>] [-profile <path>] [-dbPath <dir>] [-configPath <dir>] [-saveConfigs <boolean>]

   - **-profile <path>:** Relative path from profiles directory to the profile to be deployed. Defaults to FioranoMQ.
   - **-profilesDir <dir>:** Directory where profiles are present.
     Defaults to <fiorano_installation_dir>\fmq\profiles
   - **-configPath <dir>:** Conf directory path for the configured profile.
     Defaults to <fiorano_installation_dir>\fmq\profiles\<profile>\conf
   - **-dbPath <dir>:** Database path for configured profile.
     Defaults to <fiorano_installation_dir>\fmq\profiles\<profile>\run"
   - **-saveConfigs <boolean>:** Persists server configurations on shutdown.
     Defaults to false
   - **–help:** Displays all available options.

   Example: The command below starts the FioranoMQ HA primary server.
   fmq.bat –profile FioranoMQ_HA_rpl\HAPrimary

6. **fmq-console.bat:** This script is used to setup the class path for the Java samples included within the FioranoMQ installation.

   Usage: fmq-console.bat

   Once run, it sets the classpath for the samples and moves the command prompt to the %fiorano_installation_dir %\fmq\samples directory.

7. **fstart.bat:** This script displays a new console with the command specified as 'command line argument'. If nothing is specified it opens a new command prompt within the same current directory.

   Usage: fstart.bat **<**command**>**

   For Example: fstart.bat fmq.bat.

   This launches the FioranoMQ server with the default profile, that of FioranoMQ.

8. **recover-database.bat:** This script is used to recover the database in the event that the FioranoMQ proprietary File based datastore is corrupt.

   On launch, the script recovers the PTP database by default. For more information on the usage of the script refer to the chapter on *Fiorano DB Recovery Tool*.

   Usage: recover-database.bat [–propertiesFile <path>] [-fmq.profile <path>] [-h]

- **–propertiesFile:** Configuration file. The default configuration file is located at <fiorano_installation_dir>\fmq\profiles\recovery.properties
- **-fmq.profile <path>:** Relative path from the profiles directory to the profile expected to be deployed. Defaults to FioranoMQ.
- **-h:** Displays help topics

9. **run-client.bat:** This script is used to run Java samples included in the FioranoMQ installation. External jars can be added to classpath in fiorano_installation_dir>\fmq\bin\run-client.conf

   Usage: run-client.bat <sampleName>

10. **build-all-clients.bat**

    This script is used to compile all Java samples included in the installer. The script makes use of compile-client.sh to compile each of the samples internally. External jars can be added to classpath in <fiorano_installation_dir>/fmq/bin/complie-client.conf

    Usage: build-all-clients.bat

11. **migrate. bat and convert-database.bat**

    migrate.bat script file is used for transferring the FioranoMQ2007 database to the FioranoMQ9 format. This script uses convert-database.bat for converting the database table files and data files to FioranoMQ9 format. It is recommended that the ‘readme.txt’ file be referred to for instructions on using this utility.

12. **startCluster.bat**

    startCluster.bat script file is used for running the FioranoMQ Management server using the FioranoMQ_Clustering profile. For more information on FioranoMQ Management Server and its functions refer to Chapter 32: *Fiorano Directory Services*.

13. **routeUtility.bat**

    routeUtility.sh script file is used for creating/removing multiple routes between destinations (Queues/Topics) based on the configuration file specified on the route utility. The configuration file (routes.xml) contains the properties of each route to be created.

    A sample route configuration file can be found at $FIORANO_HOME/fmq/Utilities/RouteUtility/conf folder.

    Usage:

    $FIORANO_HOME/fmq/bin/routeUtility.sh [-operation <operation> -configFile <routeConfigsFile>]

    where,

    configFile:

    XML file that contains configurations of routes. If no configuration file is specified when the route Utility is run, then 'routes.xml' becomes the default configuration file.

    Operation:

    The operation that needs to be executed using the route utility.

Valid Values: createRoutes OR removeRoutes. If no operation is specified, 'createRoutes' is chosen by default.

14. **shutdown-fmq.bat**

This script is used to shutdown the FioranoMQ server.

This script creates a JMXConnection based on the arguments passed. If none of the arguments are passed, then default parameters are used. If the arguments passed are invalid, then an exception is displayed on the console.

Usage: shutdown-fmq.bat [-options] (For Windows)

Where options includes:

Parameter Name	**Description**	**Legal Values**
connectorType	Type of the connector which need to be created for making a jmx connection with FioranoMQ server.	RMI (Default Value), JMS
host	URL of the machine where the server is running. If the server is running on the machine as the shutdown script, the host name does not need to be specified.	localhost (Default Value), IPAddress
port	The port of the FioranoMQ server that will accept JMS connections. This value should be specified only if the connector 'type' is JMS.	1856 (Default Value)
user	Name of the user trying to shutdown the FioranoMQ server. This user should have permission to create a JMX connection.	admin (Default Value), anonymous, ayrton
passwd	Password of the user trying to shutdown the FioranoMQ Server.	passwd(Default Value for admin), anonymous (for user anonymous), senna (for user ayrton)
transportProtocol	Transport protocol which should be used for making a connection with the FioranoMQ server. This value should be specified only if the connector 'type' is JMS	TCP (Default Value), HTTP, LPC
securityProtocol	Security protocol which needs to be used to make a connection with the FioranoMQ server. By default there is no security protocol. This value should be specified only if the connector 'type' is JMS.	null (Default Value), SUN_SSL.
securityManager	Security Manager which need to be used for making a connection with FioranoMQ Server.	null(Default Value),

Parameter Name	Description	Legal Values
	By default there will not be any security Manager. This value should be specified only if the connector type is JMS	DefaultJSSESecurityManager
rmiPort	RMIConnector port on which the FioranoMQ server JMX connection is to be created. This value should be specified only if the connector type is RMI.	1858(Default value), 2059 is default value for Repeater

Example:

The commands below stop the FioranoMQ server running on a FMQWIN01 machine on default port 1856 by entering user as admin and password as passwd.

shutdown-fmq.bat -host FMQWIN01 -port 1856 -user admin -passwd passwd -transportProtocol TCP -securityProtocol null -securityManager null.

No securityManager or securityProtocol is required in this case and both are set as null. This script creates an RMIBasedJMXConnector since theconnector 'type' is not specified.

Note: If a user has configured the FioranoMQ server to accept JMSBasedJMXConnector connections, shutting down the FioranoMQ server is possible only by using this connector. Otherwise, by default, the RMIBasedJMXConnector will be used.

Scripts present under %fiorano_installation_dir %\fmq\bin\service

1. **install-fmq-service.bat:** This script launches the FioranoMQ server as an NT service. This script can be used where the FioranoMQ sever needs to start along with the system services.

   Usage: install-fmq-service.bat -profile **%PROFILE_NAME%**

   Example: install-fmq-service.bat -profile FioranoMQ_XA

FioranoMQ, when started as an NT Service, by default, takes on the FioranoMQ profile.

uninstall-fmq-service.bat: This script is used to remove the entry made in the registry of the system after install-fmq-service.bat has installed a serviceto start the FioranoMQ Server.

Usage: uninstall-fmq-service.bat -profile **%PROFILE_NAME%**

Example: uninstall-fmq-service.bat -profile FioranoMQ_XA

FioranoMQ, when started as an NT Service, by default, takes on the FioranoMQ profile.

Scripts present under <fiorano_installation_dir>\fmq\clients\c\crosscomp\scripts

1. **build_samples.bat:** This script is used for compiling all the C Cross Comp samples present under %fiorano_installation_dir %\fmq\clients\c\crosscomp\samples.

   Usage: build_samples.bat.

2. **cclientbuild.bat:** This script is used for compiling C Cross Comp samples one at a time.

   Usage: cclientbuild.bat **..\samples\<sample_type>\<sampleName>.c**

Scripts present under <fiorano_installation_dir>\fmq\clients\c\native\scripts

1. **build_samples.bat:** This script is used **for** compiling all the C Native samples present under %fiorano_installation_dir %\fmq\clients\c\native\samples

   Usage: build_samples.bat

2. **cclientbuild.bat:** This script is used to compile the C Native samples one at a time.

   Usage: cclientbuild.bat **..\samples\<sample_type>\<sampleName>.c**

Scripts present under <fiorano_installation_dir>\fmq\clients\cpp\jni\scripts

1. **build_samples.bat:** This script is used for compiling all the Cpp JNI samples present under %fiorano_installation_dir %\fmq\clients\cpp\jni\samples.

   Usage: build_samples.bat.

2. **cppclientbuild.bat:** This script is used for compiling the Cpp JNI samples, one at a time, using fmq-jni-cpprtl.lib.

   Usage: cppclientbuild.bat **..\samples\<sample_type>\<sampleName>.cpp**

3. **cppclientbuildcc.bat:** This script is used for compiling the Cpp JNI samples, one at a time, using fmq-crosscomp-cpprtl.lib.

   Usage: cppclientbuildcc.bat
..\samples\<sample_type>\<sampleName>.cpp

Scripts present under %fiorano_installation_dir %\fmq\clients\cpp\native\scripts

1. **build_samples.bat:** This script is used for compiling all the Cpp Native samples present under %fiorano_installation_dir %\fmq\clients\cpp\native\samples.

   Usage: build_samples.bat.

2. **cppclientbuild.bat:** This script is used to compile the Cpp Native samples one at a time.

   Usage: cppclientbuild.bat **..\samples\<sample_type>\<sampleName>.cpp**

Scripts present under %fiorano_installation_dir %\fmq\clients\csharp\scripts

1. **build_samples.bat:** This script is used for compiling all the unmanaged C# samples present under %fiorano_installation_dir %\fmq\clients\csharp\samples.

Usage: build_samples.bat

2. **csclientbuild.bat:** This script is used to compile the unmanaged C# samples one at a time.

   Usage: csclientbuild.bat **..\samples\<sample_type>\<sampleName>.cs**

3. **ginstall.bat:** This script is used to install the fmq-native-cpprtl.dll, fmq-native-cpprtl-https.dll, fmq-csharprtl.dll and fmq-csharprtl-https.dll dll files into the system assembly using **gacutil.exe** which is shipped with the DOT NET Installation.

   Usage: ginstall.bat

4. **vbclientbuild.bat:** This script is used for compiling the unmanaged VB samples one at a time.

   Usage: csclientbuild.bat **..\samples\<sample_type>\<sampleName>.vb**

Scripts present under %fiorano_installation_dir %\fmq\clients\nativecsharp\scripts

1. **build_samples.bat:** This script is used for compiling all the C# Native samples present under %fiorano_installation_dir %\fmq\clients\csharp\samples.

   Usage: build_samples.bat

2. **cppclientbuild.bat:** This script is used to compile the C# Native samples one at a time.

   Usage: csclientbuild.bat **..\samples\<sample_type>\<sampleName>.cs**

Scripts present under <fiorano_installation_dir>\fmq\terminal

1. **fmq-terminal.bat:** This script is used for performing admin operations through the Studio. The list of operation supported by fmq-terminal are:

   Usage: fmq-terminal.sh. The username and password are queried for logging onto the server.

   Note:

   - help command lists all the commands/operations supported by the terminal
   - <command/operation> -help displays usage of the command or operation

The list of operations supported by fmq-terminal follows:

Operation	**Description**
addMemberToGroup	adds a member to a Group
bye	exits the console
changePwd	changes password for the specified User
clearQueue	clears messages in the Queue specifed
createACF	creates the specified AdminConnectionFactory

Operation	Description
createCommonCF	creates the specified CommonConnectionFactory
createGroup	Creates the Group specified
createQCF	creates the specified QueueConnectionFactory
createQueue	creates the Queue specified Queue
createTCF	creates specified TopicConnectionFactory
createTopic	creates the Topic specified
createUser	creates new User and a new Password
createXAQCF	Creates the specified XAQueueConnectionFactory
createXATCF	creates the specified XATopicConnectionFactory
deleteACF	deletes the specified AdminConnectionFactory
deleteCommonCF	deletes the specified CommonConnectionFactory
deleteGroup	deletes the Group specified
deleteQCF	deletes the specified QueueConnectionFactory
deleteQueue	deletes the specified Queue
deleteTCF	deletes the specified TopicConnectionFactory
deleteTopic	deletes the Topic specified
deleteUser	deletes a specified User
deleteXAQCF	deletes the specified XAQueueConnectionFactory
deleteXATCF	deletes the specified XATopicConnectionFactory
exit	exits the console
help	prints the ‘help’ topic/instructions for all available commands
list	lists all available commands
listACFs	lists names of all available Admin Connection Factories
listClientIDs	lists names of all Client IDs
listCommonCFs	lists names of all available Common Connection Factories
listDomains	Lists all domains
listGroups	lists all server groups
listQCFs	lists names of all available QueueConnectionFactories
listQueues	lists names of all available Queues
listSubscriberIDs	lists SubscribterIDs of specified ClientIDs
listTCFs	lists names of all available TopicConnectionFactories
listTopics	lists names of all available Topics

Operation	Description
listUsers	lists all Users of the server
listXAQCFs	lists names of all available XAQueueConnectionFactories
listXATCFs	lists names of all available XATopicConnectionFactories
mbeanCount	prints mbeans count
quit	exits the console
restartFMQ	restarts FioranoMQ server running in the relevant Fiorano Container
run	runs specific commands in the specified file
serverInfo	prints the server information
shutdownFMQ	Shuts down the FioranoMQ server running in the relevant Fiorano Container
shutdownJBoss	Shuts down the FioranoMQ server running on the JBoss Container

Scripts present under <fiorano_installation_dir>\framework\tools\LicenseManager\bin

runLM.bat: This script is used to run the license manager. For more information on the license manager, refer to *Fiorano SOAPlatform License Manager Guide* available at: <fiorano_installation_dir>\framework\tools\LicenseManager\doc.

Scripts present under <fiorano_installation_dir>\Studio\bin

1. **InputMethodHotKey:** This script is used for configuring the InputMethod HotKey that invokes the keyboard panel to insert international characters (Japanese, Chinese, etc.).
2. **Log.bat:** This script opens the log file that stores all logs.

   Usage: Log.bat
3. **Reset.bat:** This script is used to reset all Studio logs and the Studio settings

   Usage**:** Reset.bat
4. **UserDir.bat:** This script opens the Windows Explorer at the same location as the Studio log file.

   Usage: UserDir.bat

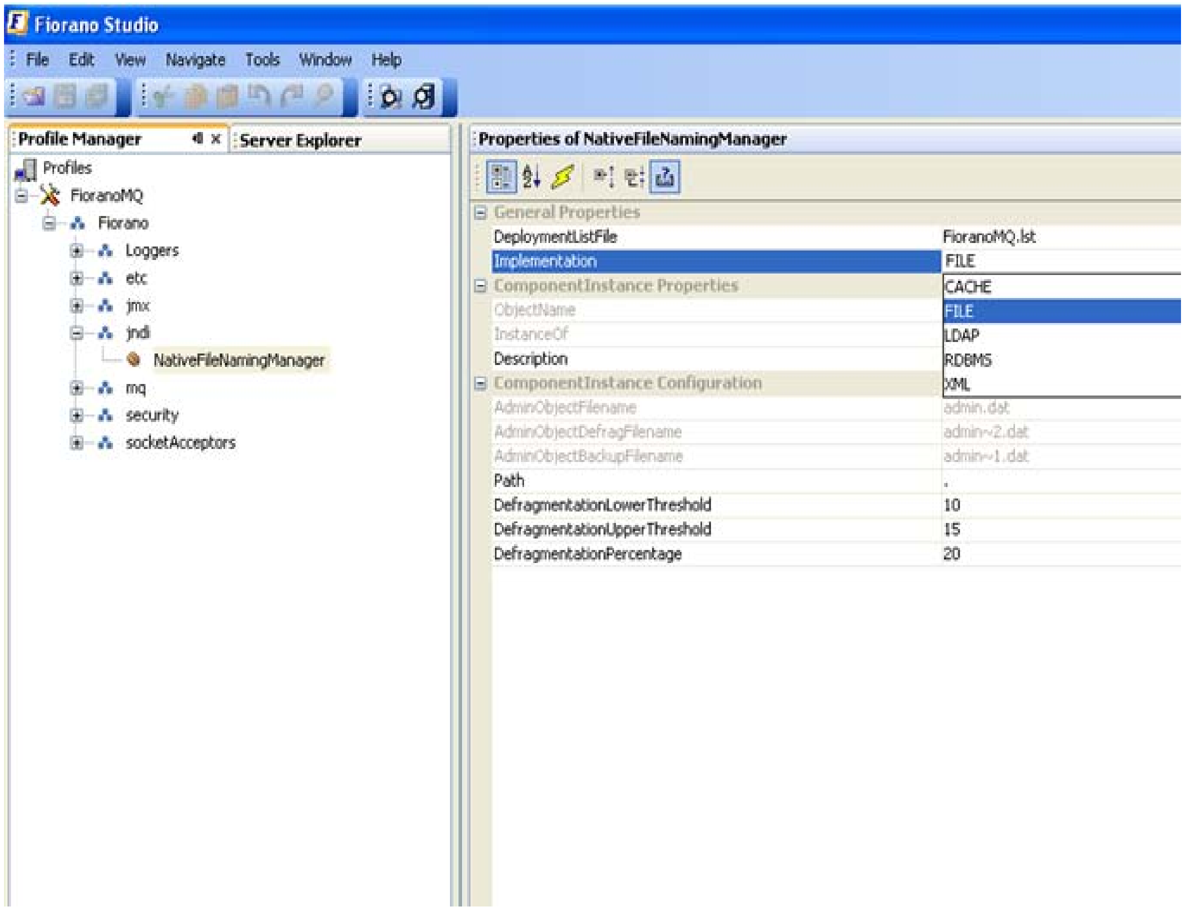
Chapter 3: Naming Manager

3.1 XML

1. Open the profile for off-line editing through the **Profile Manager** using Studio by clicking on the **Profile Manager** pane. Right-click the Profile node and select **Open Profile** from the pop-up menu. Select the desired profile and click on the **Open** button

2. Navigate to **NativeFileNamingManager** under the **jndi** node in the FioranoMQ hierarchy.

3. In the **properties panel**, change the implementation 'type' from **FILE to XML**, as shown in the figure below.

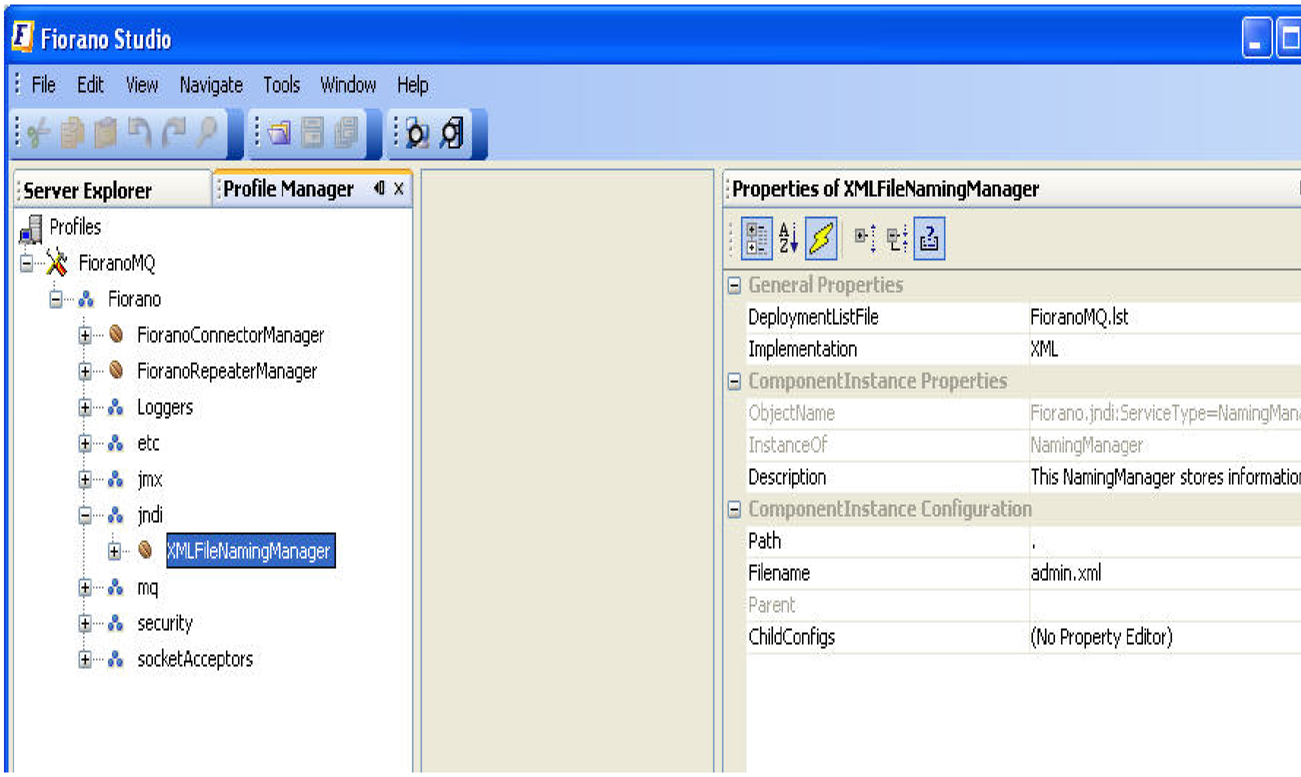


DeploymentListFile	FioranoMQ.lst
Implementation	FILE
ObjectName	
InstanceOf	
Description	
AdminObjectFilename	admin.dat
AdminObjectDefragFilename	admin~2.dat
AdminObjectBackupFilename	admin~1.dat
Path	.
DefragmentationLowerThreshold	10
DefragmentationUpperThreshold	15
DefragmentationPercentage	20

4. Right-click on the **FioranoMQ** node and select **Save** from the pop-up menu.

3.2 Configuring

1. Open the profile for off-line editing through the **Profile Manager**. Right-click the **Profiles** node and select **Open Profile** from the pop-up menu. Select the **FioranoMQ** profile and click on the **Open** button.
2. Navigate to the instance of **XMLFile NamingManager** under the **jndi** node in the FioranoMQ hierarchy.
3. In the **Properties of XMLFileNamingManager** panel, change the **Filename** property to the desired value as shown in the figure below.
4. In the **Properties of XMLFileNamingManager** change the property **Path** for the xml file.

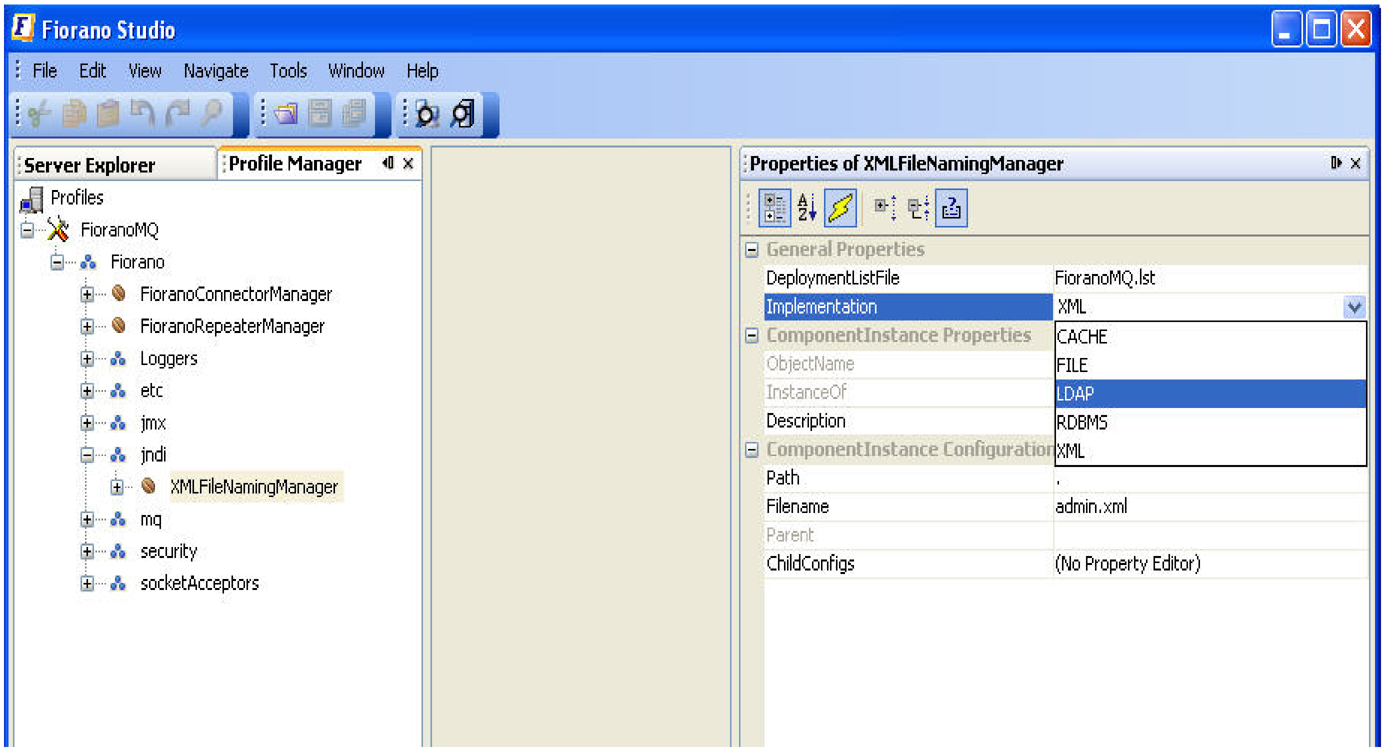


DeploymentListFile	FioranoMQ.lst
Implementation	XML
ObjectName	Fiorano.jndi:ServiceType=NamingMan
InstanceOf	NamingManager
Description	This NamingManager stores informatio
Path	.
Filename	admin.xml
Parent	
ChildConfigs	(No Property Editor)

5. Right-click on the **FioranoMQ** node and select **Save** from the pop-up menu.

3.3 LDAP

1. Open the profile for off-line editing through the **Profile Manager**. Right-click the **Profile** node and select **Open Profile** from the pop-up menu. Select the desired profile and click on the **Open** button
2. Navigate to **NativeFileNamingManager** under the **jndi** node in the FioranoMQ hierarchy.
3. In the Properties of **XMLFileNamingManager** panel, change the implementation 'type' from **FILE** to **LDAP** as shown in the figure below.

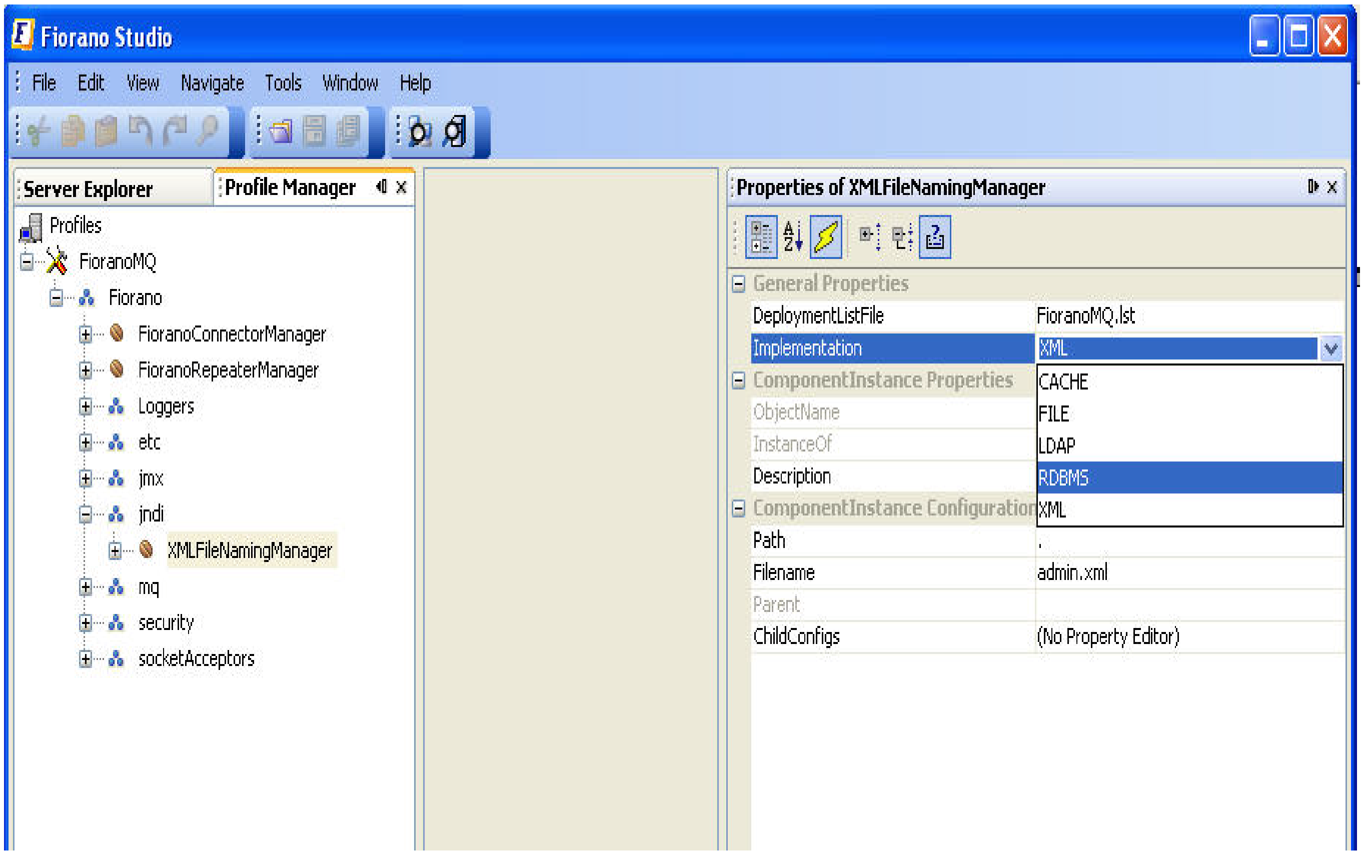


General Properties	
DeploymentListFile	FioranoMQ.lst
Implementation	XML
ComponentInstance Properties	
ObjectName	
InstanceOf	
Description	
ComponentInstance Configuration	
Path	.
Filename	admin.xml
Parent	
ChildConfigs	(No Property Editor)

4. Right-click on the **FioranoMQ** node and select **Save** from the pop-up menu.

3.4 RDBMS

1. Open the profile for off-line editing through the **Profile Manager**. Right-click the **Profile** node and select **Open Profile** from the pop-up menu. Select the desired profile and click on the **Open** button.
2. Navigate to **NativeFileNamingManager** under the jndi node in the FioranoMQ hierarchy.
3. In the **Properties of XMLFileNamingManager** pane, change the implementation 'type' from **FILE** to **RDBMS**, as shown in the figure below.
4. Right-click on the **FioranoMQ** node and select **Save** from the pop-up menu.



General Properties
DeploymentListFile: FioranoMQ.lst
Implementation: XML
ComponentInstance Properties
ObjectName
InstanceOf
Description
ComponentInstance Configuration
Path: .
Filename: admin.xml
Parent
ChildConfigs: (No Property Editor)

Chapter 4: Connection Management

This chapter discusses Connection Management in FioranoMQ. At the outset, the following points must be noted:

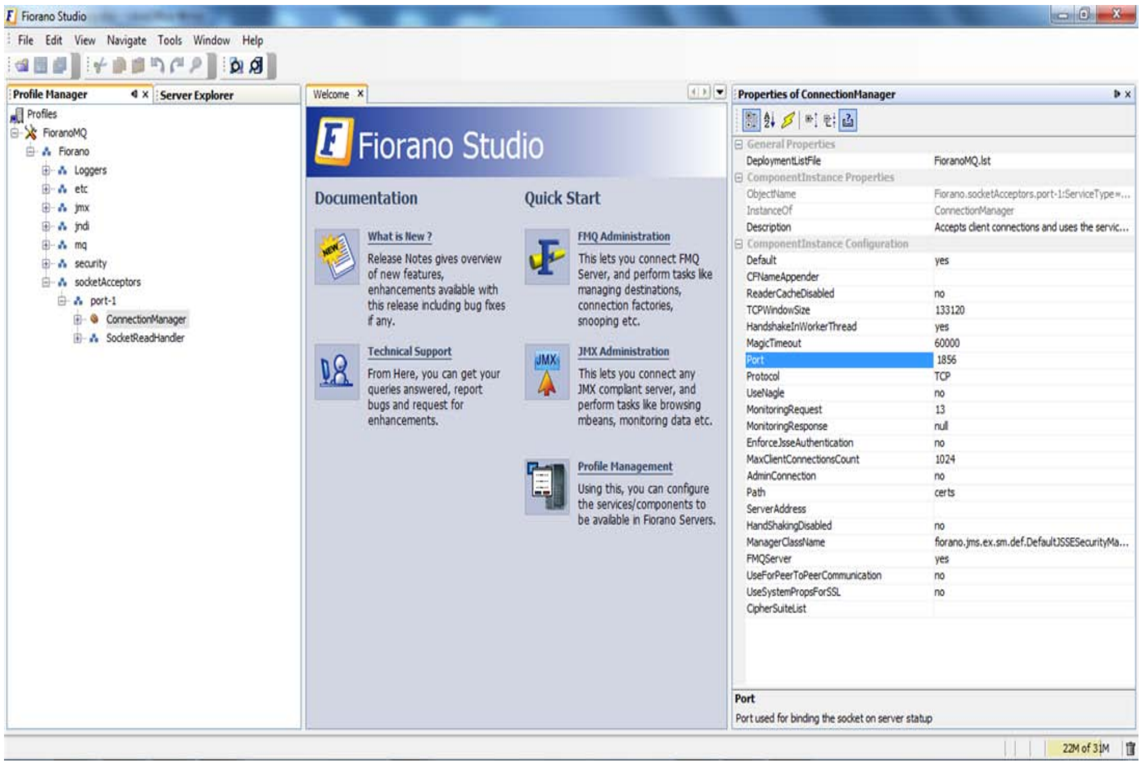
- Add the JAR files, for the database configured in the FioranoMQ Server, to the Configuration file of the server, fmq.conf, present at fmq_installation_dir\fmq\bin.
- To start the FioranoMQ Server with one of the databases, create-database.bat/sh needs to be started after adding the required JAR in the classpath of create-database.conf.

4.1 Modifying the Port Number

Note: This configuration is done in the offline mode.

For information about configuring the FioranoMQ profile through a text based file, see FioranoMQ Getting Started.

1. Launch Fiorano Studio. Select the **Profile Manager** pane and open the profile whose port number is to be changed.
2. Navigate to the **Connection Manager** instance in the profile.



What is New ?
Release Notes gives overview of new features, enhancements available with this release including bug fixes if any.

Technical Support
From Here, you can get your queries answered, report bugs and request for enhancements.

FMQ Administration
This lets you connect FMQ Server, and perform tasks like managing destinations, connection factories, snooping etc.

JMX Administration
This lets you connect any JMX compliant server, and perform tasks like browsing mbeans, monitoring data etc.

Profile Management
Using this, you can configure the services/components to be available in Fiorano Servers.

Properties of ConnectionManager

General Properties	
DeploymentListFile	FioranoMQ.lst
ComponentInstance Properties	
ObjectName	Fiorano.socketAcceptors.port-1:ServiceType=...
InstanceOf	ConnectionManager
Description	Accepts client connections and uses the servic...
ComponentInstance Configuration	
Default	yes
CFNameAppender	
ReaderCacheDisabled	no
TCPWindowSize	133120
HandshakeInWorkerThread	yes
MagicTimeout	60000
Port	1856
Protocol	TCP
UseNagle	no
MonitoringRequest	13
MonitoringResponse	null
EnforceJsseAuthentication	no
MaxClientConnectionsCount	1024
AdminConnection	no
Path	certs
ServerAddress	
HandShakingDisabled	no
ManagerClassName	fiorano.jms.ex.sm.def.DefaultJSSESecurityMa...
FMQServer	yes
UseForPeerToPeerCommunication	no
UseSystemPropsForSSL	no
CipherSuiteList	

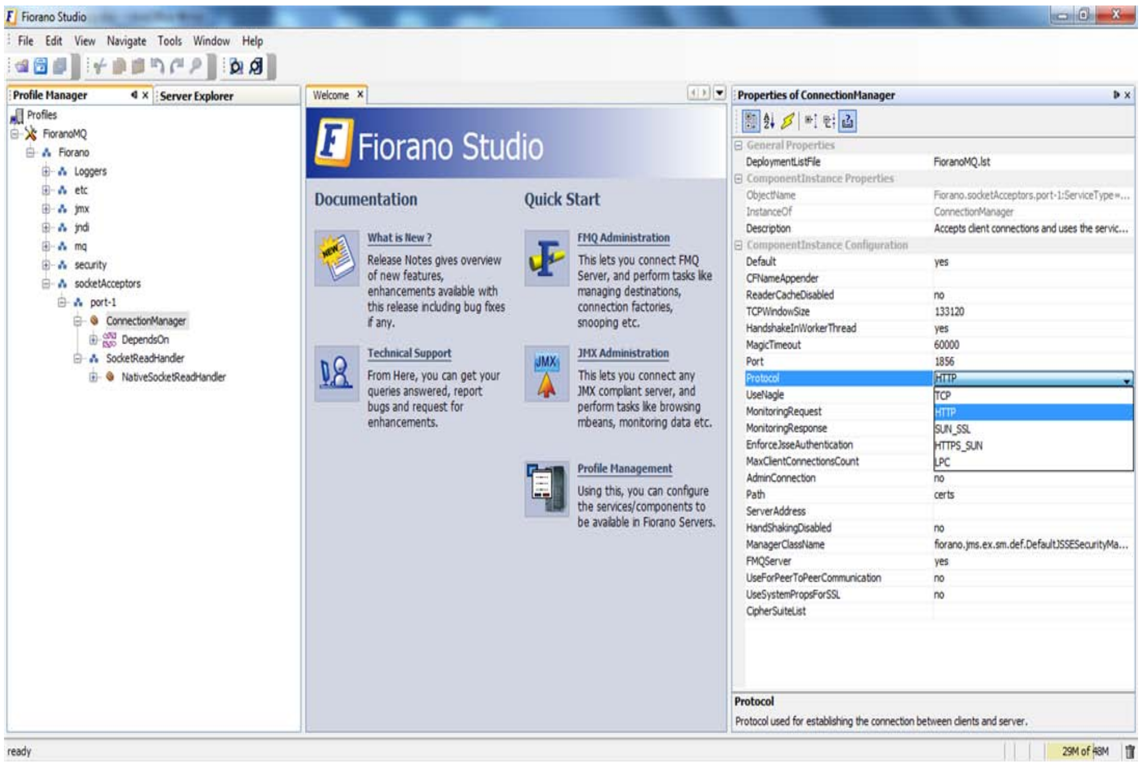
Port
Port used for binding the socket on server statup

3. Edit the port number in the **Properties Panel** as shown in the figure above.
4. Right-click on the **FioranoMQ** node and select the **Save** option from the pop-up menu.

Note: Changing port number can require the default connection factories created by the server to be re-bound.

4.2 Setting Protocol to HTTP

1. Launch the Fiorano Studio. Select the **Profile Manager** pane and open the profile whose protocol is to be set.
2. Navigate to the **Connection Manager** instance in the profile. The figure below shows the default profile.



Release Notes gives overview of new features, enhancements available with this release including bug fixes if any.

This lets you connect FMQ Server, and perform tasks like managing destinations, connection factories, snooping etc.

From Here, you can get your queries answered, report bugs and request for enhancements.

This lets you connect any JMX complaint server, and perform tasks like browsing mbeans, monitoring data etc.

Using this, you can configure the services/components to be available in Fiorano Servers.

Properties of ConnectionManager

General Properties	
DeploymentListFile	FioranoMQ.lst
ComponentInstance Properties	
ObjectName	Fiorano.socketAcceptors.port-1:ServiceType=...
InstanceOf	ConnectionManager
Description	Accepts client connections and uses the servic...
ComponentInstance Configuration	
Default	yes
CFNameAppender	
ReaderCacheDisabled	no
TCPWindowSize	133120
HandshakeInWorkerThread	yes
MagicTimeout	60000
Port	1856
Protocol	HTTP
UseNagle	TCP
MonitoringRequest	HTTP
MonitoringResponse	SUN_SSL
EnforceJsseAuthentication	HTTPS_SUN
MaxClientConnectionsCount	LPC
AdminConnection	no
Path	certs
ServerAddress	
HandShakingDisabled	no
ManagerClassName	fiorano.jms.ex.sm.def.DefaultJSSESecurityMa...
FMQServer	yes
UseForPeerToPeerCommunication	no
UseSystemPropsForSSL	no
CipherSuiteList	

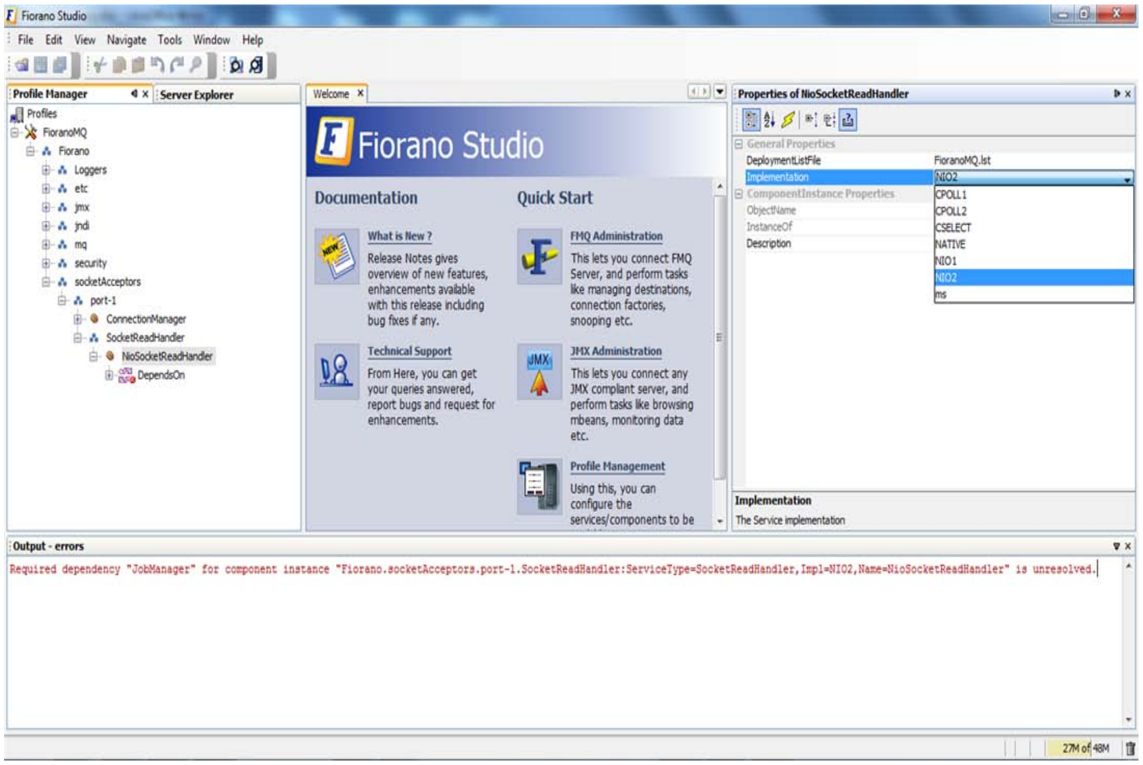
Protocol
Protocol used for establishing the connection between clients and server.

3. Set the protocol to **HTTP** from the drop-down list in the **Properties Panel** as shown in the figure above.
4. Right-click the FioranoMQ node and select the **Save** option from the pop-up menu.

Note: Changing protocol can require the default connection factories created by the server to be re-bound.

4.3 Modifying the Thread Management Policy

1. Launch the Fiorano Studio. Open the **Profile Manager** and open the profile whose thread management policy is to be modified.
2. Navigate to the **Connection Manager** instance in the profile. As with the default profile, this is at the node **Fiorano -> socketAcceptors -> port-1 -> Connection Manager**.
3. From the list of dependencies, right-click on **SocketReadHandler** and select Locate in Tree to reach the instance of SocketReadHandler used by the connection manager.
4. In the Properties of the **NativeSocketReadHandler** pane, select **NIO2** from the drop-down list of the **Implementation** options, as shown in the figure below.


Fiorano Studio
File Edit View Navigate Tools Window Help
Profile Manager
Server Explorer
Profiles
FioranoMQ
Fiorano
Loggers
etc
jmx
jndi
mq
security
socketAcceptors
port-1
ConnectionManager
SocketReadHandler
NioSocketReadHandler
DependsOn
Welcome
Fiorano Studio
Documentation
Quick Start
What is New ?
Release Notes gives overview of new features, enhancements available with this release including bug fixes if any.
FMQ Administration
This lets you connect FMQ Server, and perform tasks like managing destinations, connection factories, snooping etc.
Technical Support
From Here, you can get your queries answered, report bugs and request for enhancements.
JMX Administration
This lets you connect any JMX compliant server, and perform tasks like browsing mbeans, monitoring data etc.
Profile Management
Using this, you can configure the services/components to be
27M of 48M

Properties of NioSocketReadHandler

General Properties

DeploymentListFile	FioranoMQ.lst
Implementation	NIO2

ComponentInstance Properties

ObjectName	
InstanceOf	
Description	

CPOLL1
CPOLL2
CSELECT
NATIVE
NIO1
NIO2
ms

Implementation
The Service implementation

Output - errors

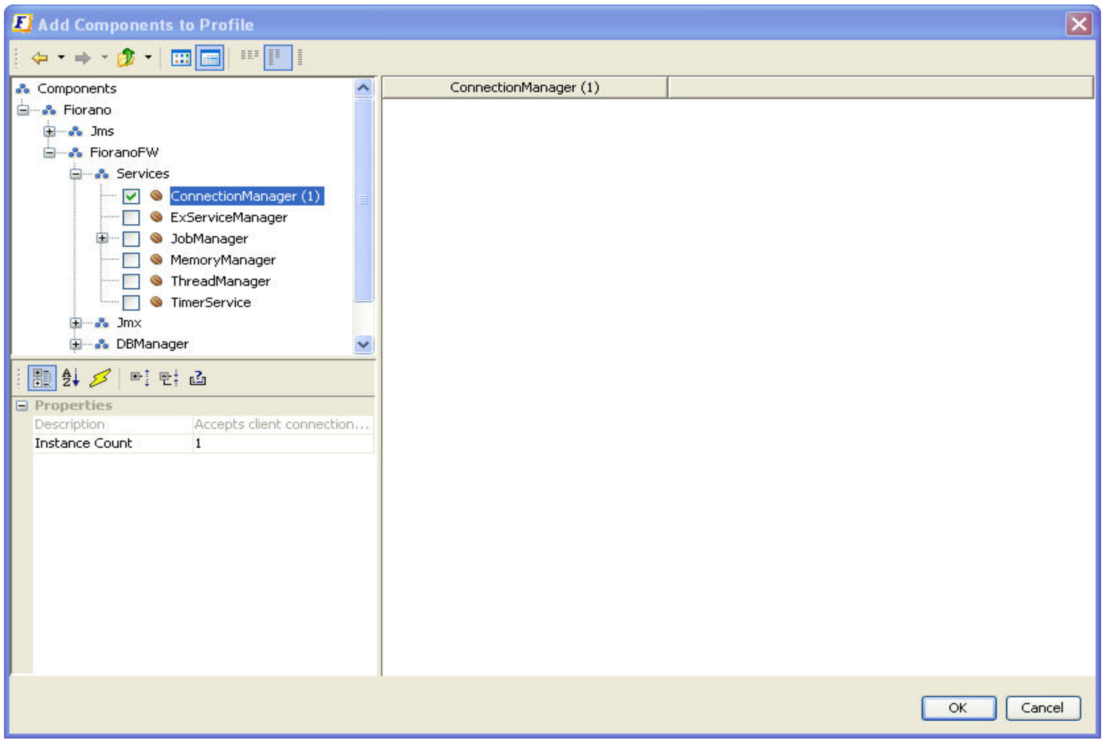
```
Required dependency "JobManager" for component instance "Fiorano.socketAcceptors.port-1.SocketReadHandler:ServiceType=SocketReadHandler,Impl=NIO2,Name=NioSocketReadHandler" is unresolved.
```

5. Right-click the **FioranoMQ** node and select the **Save** option from the pop-up menu. Then run the server.

4.4 Adding a Socket Acceptor

Note: This configuration is done in the offline mode

1. Launch the Fiorano Studio. Open the **Profile Manager** and open the profile where the Socket Acceptor is to be added.
2. In the profile, select the domain to which the new Socket Acceptor is added. The default FioranoMQ profile has socket acceptors at the following nodes in the tree **Fiorano -> socketAcceptors**. Fiorano recommends adding a new sub-domain (for instance, port-2) to this domain along with a new socket acceptor.
3. Right-click on the desired domain and select **Add Components**. The **Add Components to Profile** dialog box appears. Navigate to **Fiorano -> FioranoFw -> Services**
4. Select the component **Connection Manager1** from the new dialog box.



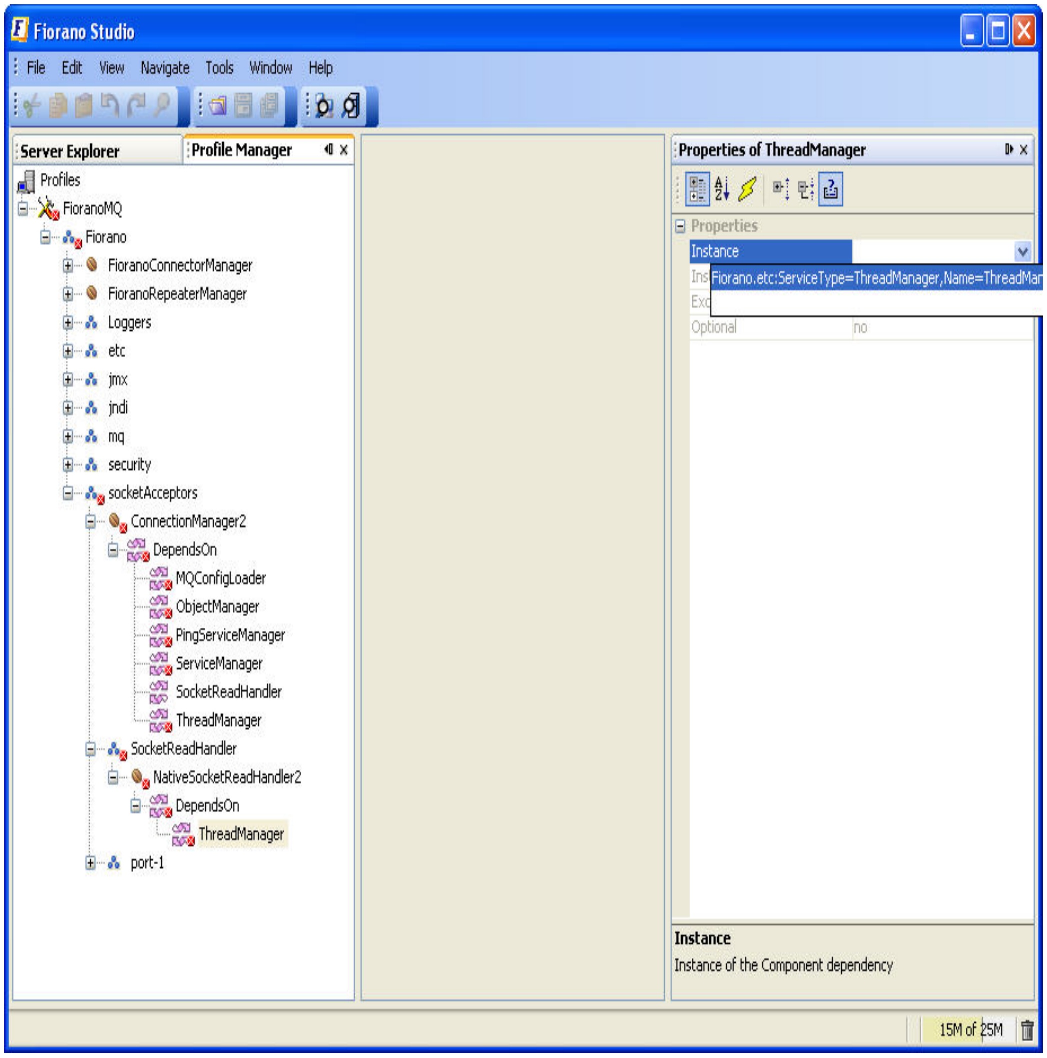
Properties	
Description	Accepts client connection...
Instance Count	1

Note: More than one instance can be added by specifying the desired Instance Count in the properties panel.

5. Click the **OK** button to add the selected instance(s) to the profile. The dependencies of the newly added component(s) have to be resolved. All un-resolved dependencies are marked with an error icon which is red in color.

   Note: In addition to the Manager instance, an instance of the SocketReadHandler is added to the profile as well.

6. To resolve dependencies, open the **DependsOn** property of the newly added Connection Manager and the associated SocketReadHandler. For each dependency marked with a red colored icon, select the desired instance from the drop-down list from the **Properties** field.



Note: Any existing instance for a dependency could be used to resolve it.

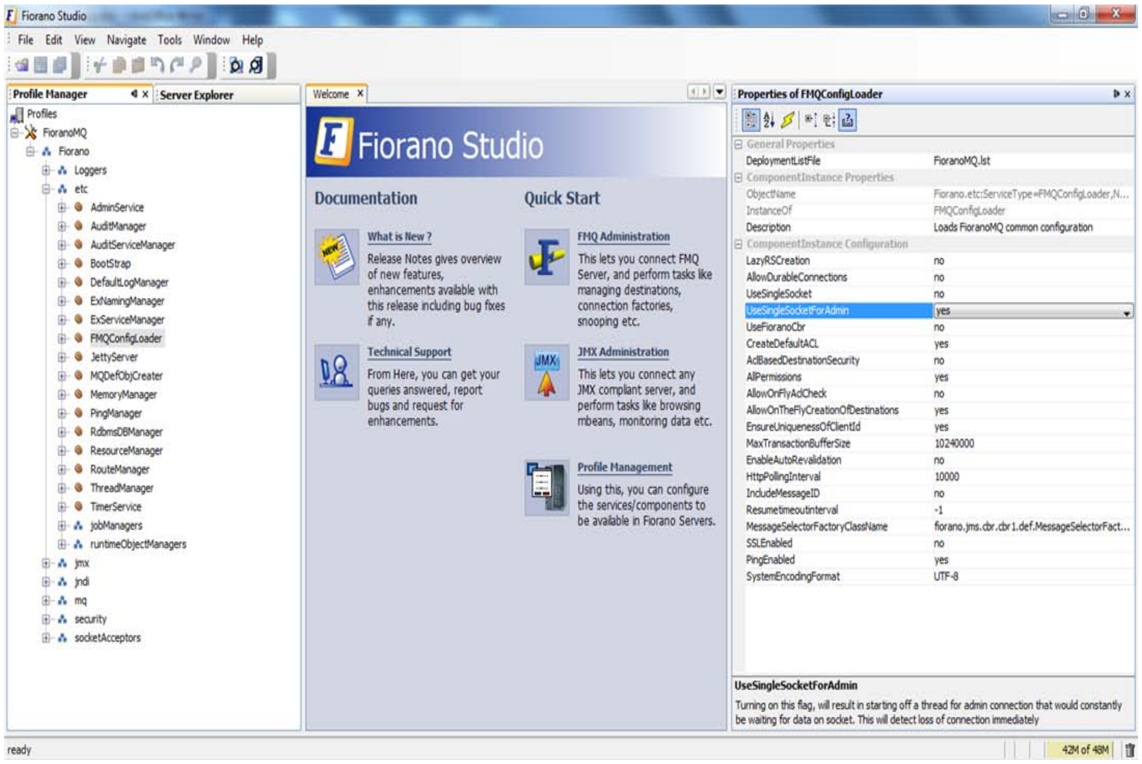
7. Right-click on the profile root node and select **Validate** to ensure that all dependencies are resolved.
8. Modify the port number for the newly added Socket Acceptor so that it is different from the port number in use by existing socket acceptor.
9. Right-click on the profile root node and select **Save** to save the profile.

4.4.1 Configuring Single Socket Acceptor for Admin

Note: This configuration can be done only in offline mode.

1. Open the **Studio** and open the **FiranoMQ** profile.

2. Navigate to **FioranoMQ -> Fiorano -> etc -> FMQConfigLoader -> UseSingleSocketForAdmin.**



Release Notes gives overview of new features, enhancements available with this release including bug fixes if any.

This lets you connect FMQ Server, and perform tasks like managing destinations, connection factories, snooping etc.

From Here, you can get your queries answered, report bugs and request for enhancements.

This lets you connect any JMX compliant server, and perform tasks like browsing mbeans, monitoring data etc.

Using this, you can configure the services/components to be available in Fiorano Servers.

Properties of FMQConfigLoader

General Properties	
DeploymentListFile	FioranoMQ.lst
ComponentInstance Properties	
ObjectName	Fiorano.etc:ServiceType=FMQConfigLoader,N...
InstanceOf	FMQConfigLoader
Description	Loads FioranoMQ common configuration
ComponentInstance Configuration	
LazyRSCreation	no
AllowDurableConnections	no
UseSingleSocket	no
UseSingleSocketForAdmin	yes
UseFioranoCbr	no
CreateDefaultACL	yes
AclBasedDestinationSecurity	no
AllPermissions	yes
AllowOnFlyAclCheck	no
AllowOnTheFlyCreationOfDestinations	yes
EnsureUniquenessOfClientId	yes
MaxTransactionBufferSize	10240000
EnableAutoRevalidation	no
HttpPollingInterval	10000
IncludeMessageID	no
Resumetimeoutinterval	-1
MessageSelectorFactoryClassName	fiorano.jms.cbr.cbr1.def.MessageSelectorFact...
SSLEnabled	no
PingEnabled	yes
SystemEncodingFormat	UTF-8

UseSingleSocketForAdmin

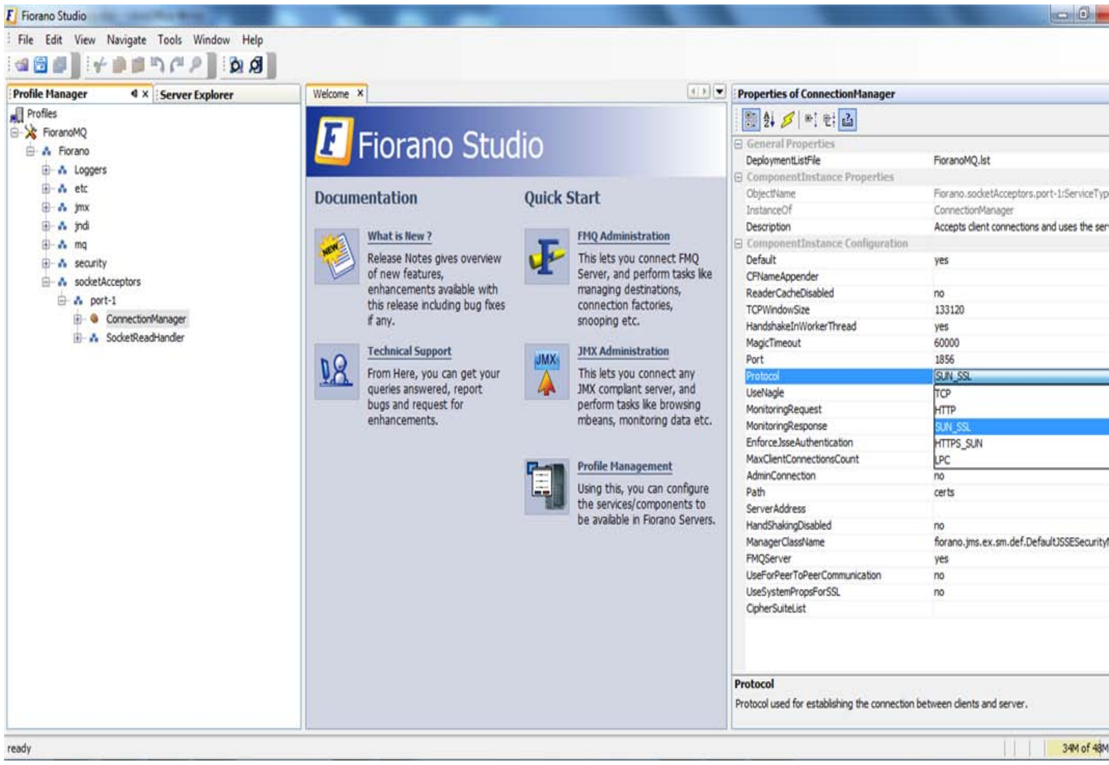
Turning on this flag, will result in starting off a thread for admin connection that would constantly be waiting for data on socket. This will detect loss of connection immediately

By default the value is set to '**yes'**. Turning on this flag results in starting a thread for admin connection that would wait for data on the socket. Any loss of connection is detected immediately.

4.5 Enabling SSL in FioranoMQ Messaging Server

1. TCP with JSSE Security

- Launch the **Fiorano Studio** for offline configuration of the FioranoMQ server.
- Select **Tools > Configure Profile** from the menu bar and open the profile needed. Navigate to **%selectedProfile% > Fiorano > SocketAcceptors > Port1 > ConnectionManager** in the **Server Explorer**.



What is New ?
Release Notes gives overview of new features, enhancements available with this release including bug fixes if any.

Technical Support
From Here, you can get your queries answered, report bugs and request for enhancements.

FMQ Administration
This lets you connect FMQ Server, and perform tasks like managing destinations, connection factories, snooping etc.

JMX Administration
This lets you connect any JMX compliant server, and perform tasks like browsing mbeans, monitoring data etc.

Profile Management
Using this, you can configure the services/components to be available in Fiorano Servers.

Properties of ConnectionManager

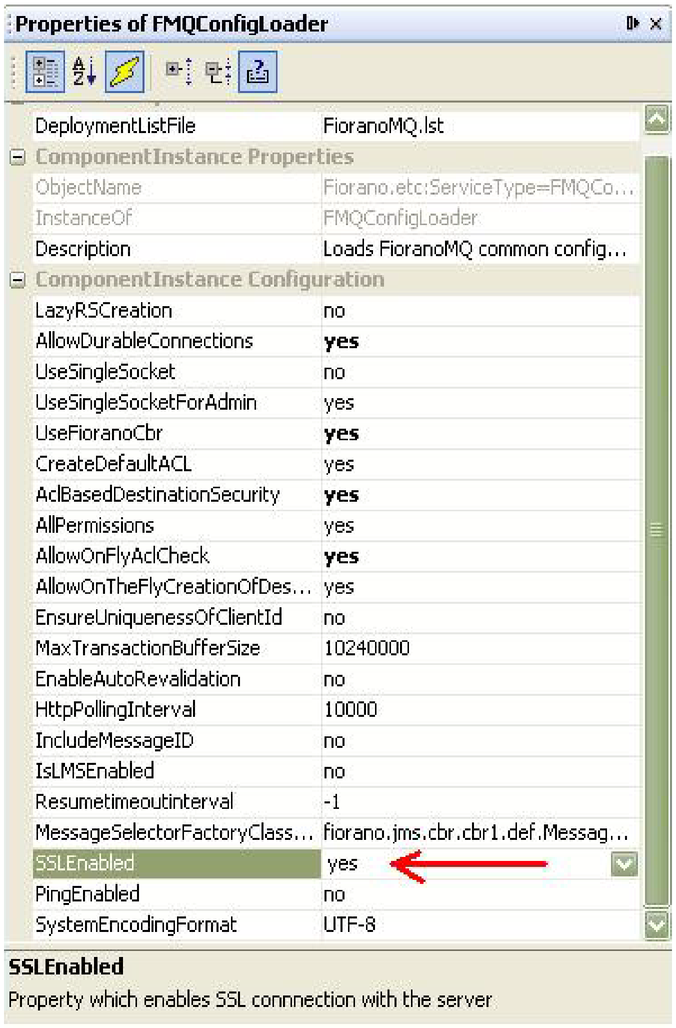
General Properties	
DeploymentListFile	FioranoMQ.lst
ComponentInstance Properties	
ObjectName	Fiorano.socketAcceptors.port-1:ServiceTyp
InstanceOf	ConnectionManager
Description	Accepts client connections and uses the ser
ComponentInstance Configuration	
Default	yes
CFNameAppender	
ReaderCacheDisabled	no
TCPWindowSize	133120
HandshakeInWorkerThread	yes
MagicTimeout	60000
Port	1856
Protocol	SUN_SSL
UseNagle	TCP
MonitoringRequest	HTTP
MonitoringResponse	SUN_SSL
EnforceJsseAuthentication	HTTPS_SUN
MaxClientConnectionsCount	LPC
AdminConnection	no
Path	certs
ServerAddress	
HandShakingDisabled	no
ManagerClassName	fiorano.jms.ex.sm.def.DefaultJSSESecurityM
FMQServer	yes
UseForPeerToPeerCommunication	no
UseSystemPropsForSSL	no
CipherSuiteList	

Protocol
Protocol used for establishing the connection between clients and server.

- Change the **Protocol** property to **SUN_SSL**.
- Change the **UseSystemPropsForSSL** to **true** (Optional)

Note : The public/private keys and/or certificates used by the FioranoMQ Server can be loaded by specifying the related system properties or by installing the appropriate security managers which can load the certificates. Please see the note at the start of Section 4.5.1

- Navigate to **%selectedProfile% > Fiorano > etc > FMQConfigLoader**. Right-click on **FMQConfigLoader** and select **Add Attribute** from the pop-up menu. Add an additional attribute with the name **SSLEnabled** and with a value that is 'true'.



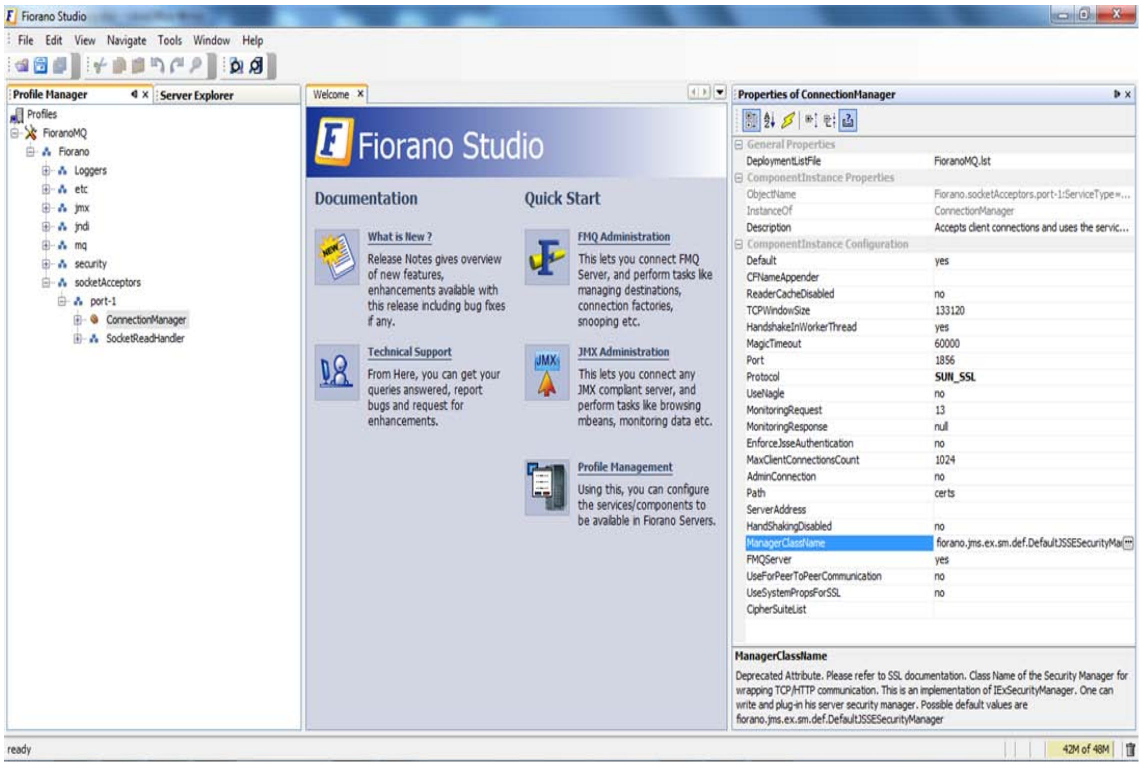
Properties of FMQConfigLoader

DeploymentListFile	FioranoMQ.lst
ComponentInstance Properties	
ObjectName	Fiorano.etc:ServiceType=FMQCo...
InstanceOf	FMQConfigLoader
Description	Loads FioranoMQ common config...
ComponentInstance Configuration	
LazyRSCreation	no
AllowDurableConnections	**yes**
UseSingleSocket	no
UseSingleSocketForAdmin	yes
UseFioranoCbr	**yes**
CreateDefaultACL	yes
AclBasedDestinationSecurity	**yes**
AllPermissions	yes
AllowOnFlyAclCheck	**yes**
AllowOnTheFlyCreationOfDes...	yes
EnsureUniquenessOfClientId	no
MaxTransactionBufferSize	10240000
EnableAutoRevalidation	no
HttpPollingInterval	10000
IncludeMessageID	no
IsLMSEnabled	no
Resumetimeoutinterval	-1
MessageSelectorFactoryClass...	fiorano.jms.cbr.cbr1.def.Messag...
SSLEnabled	yes
PingEnabled	no
SystemEncodingFormat	UTF-8

SSLEnabled
Property which enables SSL connnection with the server

- Navigate to **%selectedProfile% > Fiorano > socketAcceptors > port-1 > ConnectionManager.** Check the default value of property **ManagerClassName**. Ensure that the default value of ManagerClassName is **fiorano.jms.ex.sm.def.DefaultJSSESecurityManager**. (Optional)

Note : This parameter is deprecated. Alternatively, in order to load the KeyStore and TrustStore for initializing the context in which SSL Sockets are created, corresponding system properties should be set and **UseSystemPropsForSSL** should be set to true.



Release Notes gives overview of new features, enhancements available with this release including bug fixes if any.

This lets you connect FMQ Server, and perform tasks like managing destinations, connection factories, snooping etc.

From Here, you can get your queries answered, report bugs and request for enhancements.

This lets you connect any JMX compliant server, and perform tasks like browsing mbeans, monitoring data etc.

Using this, you can configure the services/components to be available in Fiorano Servers.

Properties of ConnectionManager

General Properties	
DeploymentListFile	FioranoMQ.lst
ComponentInstance Properties	
ObjectName	Fiorano.socketAcceptors.port-1:ServiceType=...
InstanceOf	ConnectionManager
Description	Accepts client connections and uses the servic...
ComponentInstance Configuration	
Default	yes
CFNameAppender	
ReaderCacheDisabled	no
TCPWindowSize	133120
HandshakeInWorkerThread	yes
MagicTimeout	60000
Port	1856
Protocol	**SUN_SSL**
UseNagle	no
MonitoringRequest	13
MonitoringResponse	null
EnforceJsseAuthentication	no
MaxClientConnectionsCount	1024
AdminConnection	no
Path	certs
ServerAddress	
HandShakingDisabled	no
ManagerClassName	fiorano.jms.ex.sm.def.DefaultJSSESecurityMa
FMQServer	yes
UseForPeerToPeerCommunication	no
UseSystemPropsForSSL	no
CipherSuiteList	

ManagerClassName

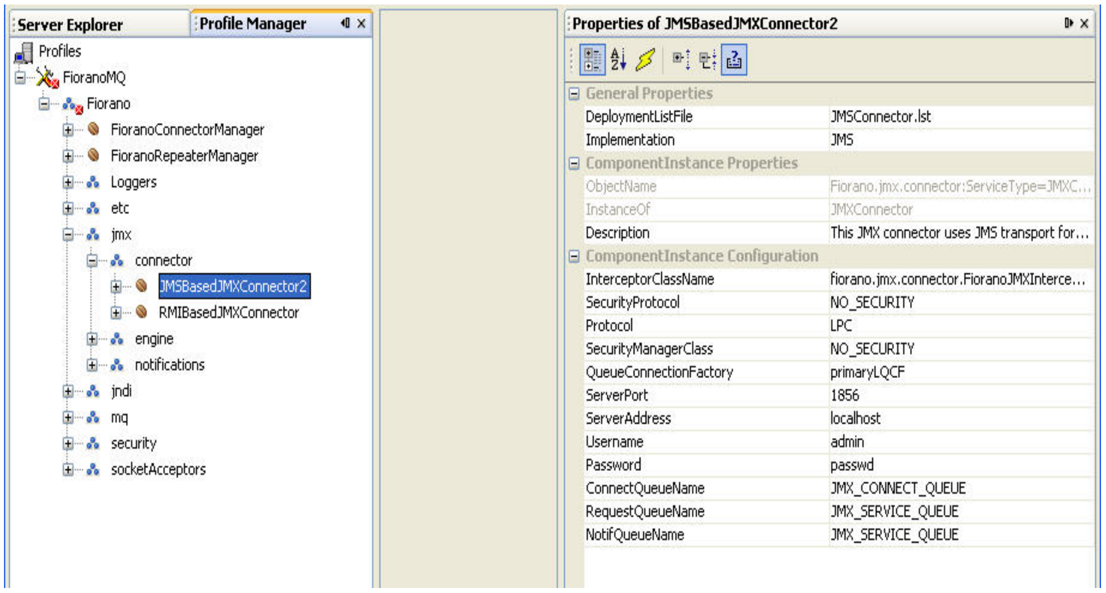
Deprecated Attribute. Please refer to SSL documentation. Class Name of the Security Manager for wrapping TCP/HTTP communication. This is an implementation of IExSecurityManager. One can write and plug-in his server security manager. Possible default values are fiorano.jms.ex.sm.def.DefaultJSSESecurityManager

- Navigate to **Fiorano > jmx > connector > JMSBasedJMXConnector2** and set the following properties to allow the JMSConnector to connect to the secure server.

    a. **Protocol:** TCP

    b. **SecurityManagerClass:** fiorano.jmx.connector.fmq.security.JSSESecurityManager

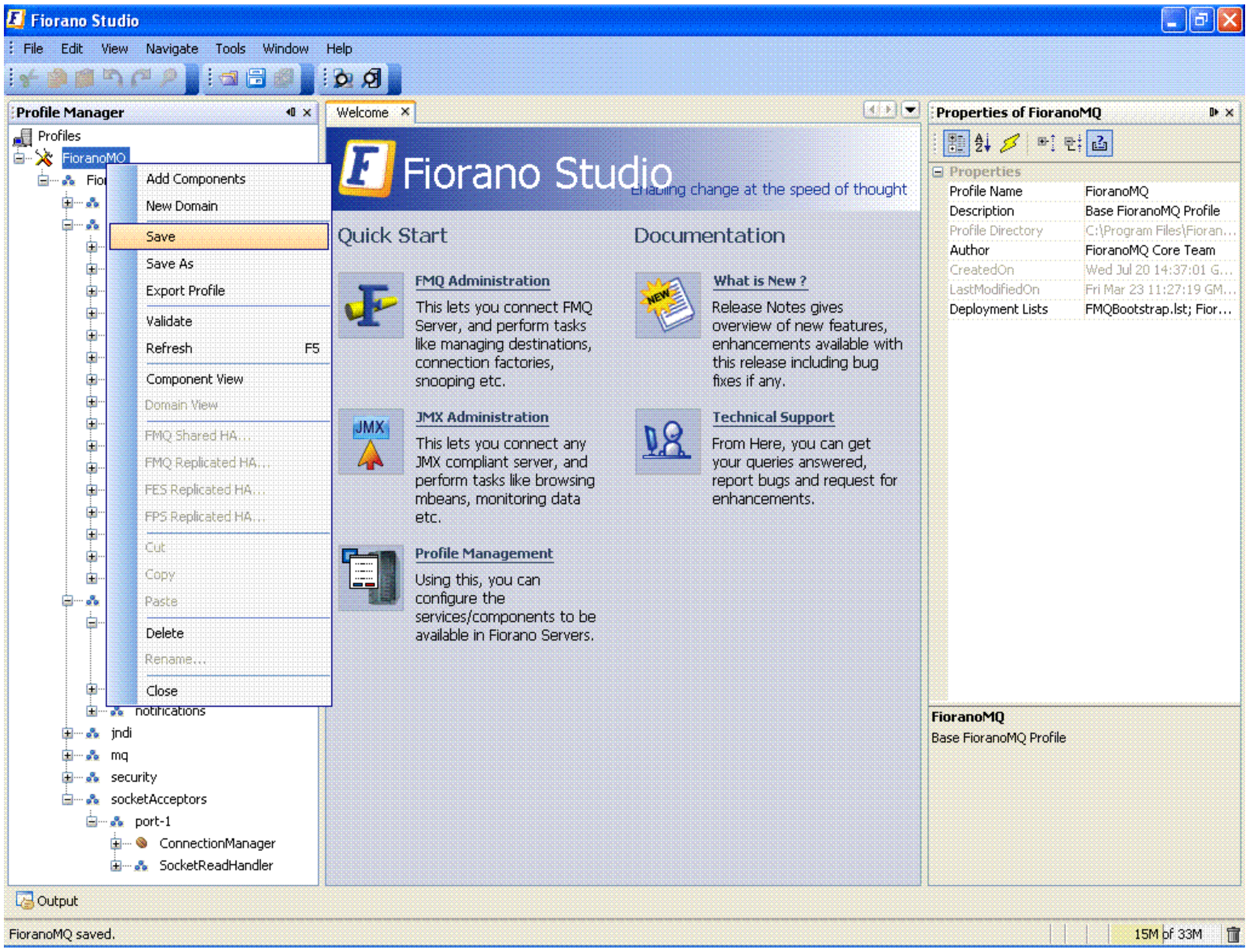
    c. **SecurityProtocol:** SUN_SSL



Properties of JMSBasedJMXConnector2

General Properties	
DeploymentListFile	JMSConnector.lst
Implementation	JMS
ComponentInstance Properties	
ObjectName	Fiorano.jmx.connector:ServiceType=JMXC...
InstanceOf	JMXConnector
Description	This JMX connector uses JMS transport for...
ComponentInstance Configuration	
InterceptorClassName	fiorano.jmx.connector.FioranoJMXInterce...
SecurityProtocol	NO_SECURITY
Protocol	LPC
SecurityManagerClass	NO_SECURITY
QueueConnectionFactory	primaryLQCF
ServerPort	1856
ServerAddress	localhost
Username	admin
Password	passwd
ConnectQueueName	JMX_CONNECT_QUEUE
RequestQueueName	JMX_SERVICE_QUEUE
NotifQueueName	JMX_SERVICE_QUEUE

- Right-click the **FioranoMQ** domain in the **Profile Manager** and select the **Save** option from the pop-up menu. Changes are saved in the *Configs.xml* file.


Fiorano Studio
File Edit View Navigate Tools Window Help
Profile Manager
Profiles
FioranoMQ
Add Components
New Domain
Save
Save As
Export Profile
Validate
Refresh F5
Component View
Domain View
FMQ Shared HA...
FMQ Replicated HA...
FES Replicated HA...
FPS Replicated HA...
Cut
Copy
Paste
Delete
Rename...
Close
notifications
jndi
mq
security
socketAcceptors
port-1
ConnectionManager
SocketReadHandler
Welcome
Fiorano Studio
Enabling change at the speed of thought
Quick Start
Documentation
FMQ Administration
This lets you connect FMQ Server, and perform tasks like managing destinations, connection factories, snooping etc.
What is New ?
Release Notes gives overview of new features, enhancements available with this release including bug fixes if any.
JMX Administration
This lets you connect any JMX compliant server, and perform tasks like browsing mbeans, monitoring data etc.
Technical Support
From Here, you can get your queries answered, report bugs and request for enhancements.
Profile Management
Using this, you can configure the services/components to be available in Fiorano Servers.
Properties of FioranoMQ
Properties
Output
15M of 33M

Profile Name	FioranoMQ
Description	Base FioranoMQ Profile
Profile Directory	C:\Program Files\Fioran...
Author	FioranoMQ Core Team
CreatedOn	Wed Jul 20 14:37:01 G...
LastModifiedOn	Fri Mar 23 11:27:19 GM...
Deployment Lists	FMQBootstrap.lst; Fior...

FioranoMQ
Base FioranoMQ Profile

FioranoMQ saved.

- Clear the existing database using script ClearDB.bat located in %FIORANO_HOME%\fmq\bin directory.

  ClearDB.bat %selectedProfile%

- Start the Server again using script file fmq.bat located in %FIORANO_HOME%\fmq\bin directory.

  fmq.bat –profile %selectedProfile%

The server starts accepting connections on TCP in the SSL (JSSE) mode.

2. HTTP with JSSE Security

1. Launch the **Fiorano Studio** for offline configuration of the FioranoMQ server.
2. Select **Tools > Configure Profile** from the menu bar, and open the profile needed. Navigate to **%selectedProfile% > Fiorano > SocketAcceptors > Port1 > ConnectionManager**.
3. Change the protocol property from **TCP** to **HTTPS_SUN**.
4. Navigate to **%selectedProfile% > Fiorano > etc > FMQConfigLoader**. Right-click on **FMQConfigLoader** and select **Add Attribute** from the pop-up menu. Add an additional attribute with the name **SSLEnabled** and with the value '**true'**.
5. Navigate to **%selectedProfile% > Fiorano > socketAcceptors > port-1 > ConnectionManager**. Check the default value of property ManagerClassName. Ensure that the default value of Security manager is **fiorano.jms.ex.sm.def.DefaultJSSESecurityManager**.
6. Navigate to **Fiorano > jmx > connector > JMSBasedJMXConnector2** and set the following properties to allow JMSConnector to connect to the secure server:
   a. **Protocol:** HTTP
   b. **SecurityManagerClass:** fiorano.jmx.connector.fmq.security.JSSESecurityManager
   c. **SecurityProtocol:** SUN_SSL
7. Right-click the **FioranoMQ** domain in the **Server Explorer** and select the **Save** option from the pop-up menu. Changes are saved in the Configs.xml file.
8. Clear the existing database using script ClearDB.bat located in the %FIORANO_HOME%\fmq\bin directory.

   ClearDB.bat %selectedProfile%
9. Start the Server again using script file fmq.bat located in %FIORANO_HOME%\fmq\bin directory.

   fmq.bat –profile %selectedProfile%

The server starts accepting connections on HTTPS in the SSL (JSSE) mode.

Note:

1. To enabling SSL for FES/FPS servers, steps taken are similar to the ones above. For FPS profiles the Protocol, SecurityProtocol and SecurityManagerClass properties for Fiorano > jmx >Engine > ClientJMXEngine also need to be changed.
2. When FioranoMQ server is running with HTTPS_SUN protocol, pinging is enabled at the server. Also, the client connecting to server must enable ping.

4.5.1 Starting FMQ Server in SSL Mode

The basic configurations for the FMQ server consist of changing the default TCP protocol to 'SUN_SSL' in the connection manager properties and turning on the SSL flag in the FMQConfigLoader.

In order to use System properties to specify the KeyStore, TrustStore and the corresponding password which are used to load and initialize the "context" in which the SSL-enabled Server Socket is created, enable **UseSystemPropsForSSL** parameter in the Connection Manager configuration by following the similar steps followed to configure Port Number in Section 4.1 . If this parameter is enabled, then the following system properties should be set before starting the FioranoMQ Server:

- javax.net.ssl.keyStore
- javax.net.ssl.keyStorePassword
- javax.net.ssl.keyStoreType (Optional, if not set JDK default will be used)
- javax.net.ssl.trustStore
- javax.net.ssl.trustStorePassword
- javax.net.ssl.trustStoreType (Optional, if not set JDK default will be used)
- javax.net.debug (Optional, if not set JDK default will be used)

These system properties should be specified in the file - "%INSTALLER_HOME%/fmq/bin/fmq.conf" - under the section <java.system.props>

Alternatively, if **UseSystemPropsForSSL** is not enabled, by default, the FMQ server uses 'DefaultJSSEKeys', the Java keystore file for authenticating the client connections. This keyStore is loaded in the Client or Server process using the Security Managers installed for client and server respectively. Other certificate files can be ignored as they are not in used. To check which certificates are in a Java keystore, the 'keytool' utility which comes along with the JDK installed is used. To check which certificates are stored in the Java keystore 'DefaultJSSEKeys' file the following command can be used:

```
keytool  -list  -v  -keystore  DefaultJSSEKeys
```

Note: 'keytool' can be found at %JAVA_HOME%\bin\keytool

4.5.2 Generating 'Keystores' of 'type' JKS, provided by SUN

Java Keytool is a key and certificate management utility. It allows users to manage their own public/private key pairs and certificates. It also allows users to cache certificates. Java Keytool stores the keys and certificates in a 'keystore'.

The following command can be used to create a 'keystore':

```
keytool -genkey -alias sampleKS -keyalg RSA -keystore C:\keystore
```

Where,

- -‘genkey’- is used to generate a key pair (a public key and associated private key). This wraps the public key into an X.509 v1 self-signed certificate that is stored as a single-element certificate chain. This certificate chain and the private key are stored in a new keystore entry identified by an alias.
- -alias - is the unique alias for accessing keystore entries (key and trusted certificate entries). Typically, the company name or hostname of the Server forms the alias.
- -keyalg - specifies the algorithm to be used to create the key pair.
- -keystore - is used for specifying the name and location of the persistent keystore file for keystore that is managed by the keytool.

Note: keytool –help to reveal other options of keytool.

Once the above command is executed for creating the certificate, information related to the creation of the certificate needs to be provided as shown below:

Enter keystore password: passwd

What is your first and last name?

[Unknown]: John

What is the name of your organizational unit?

[Unknown]: FMQ

What is the name of your organization?

[Unknown]: Fiorano

What is the name of your City or Locality?

[Unknown]: Los Gatos

What is the name of your State or Province?

[Unknown]: CA

What is the two-letter country code for this unit?

[Unknown]: US

Is CN=John, OU=MQ, O=Fiorano, L= Los Gatos, ST= CA, C=US correct?

[no]: yes

Enter key password for <sampleKS>

(RETURN if same as keystore password):

This keystore acts as a self-signed certificate. A certificate request can be generated in the event that the certificate needs to be signed by Certification Authorities such as Verisign or eTrust.

4.5.3 Server Side Configurations

The Security Manager loads the certificates/keys from the Keystore that has been created. Please note that this section describes the procedure to create and install the Server security manager, only when **UseSystemPropsForSSL** is disabled. In case, this property is enabled, this section can be skipped and the KeyStore and TrustStore location and related properties can be provided directly as system properties for the FioranoMQ Server.

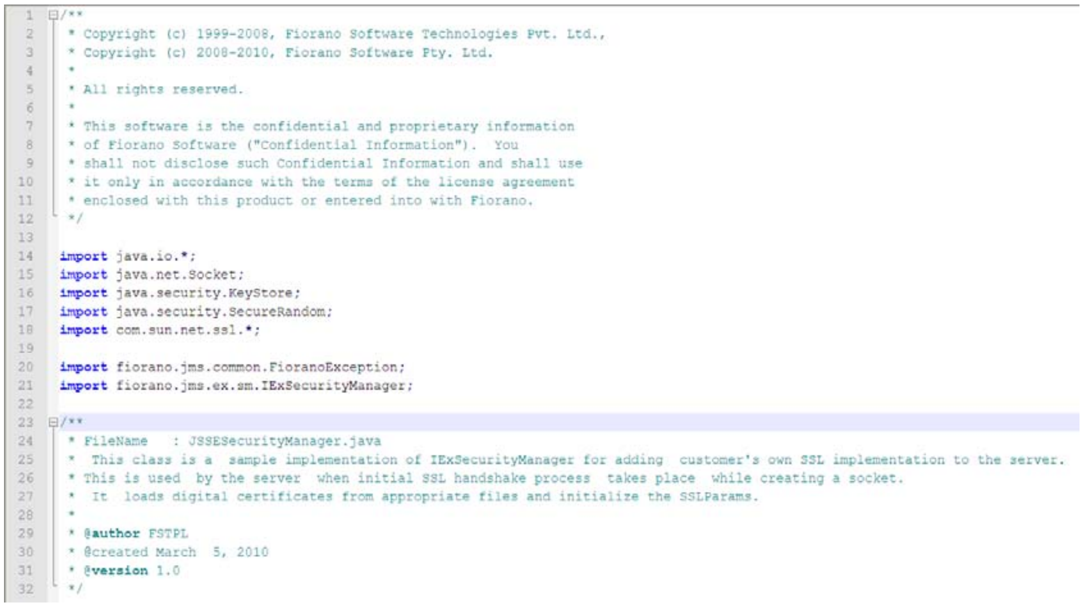
Security manager class should implement **fiorano.jms.ex.sm.IExSecurityManager**.

The server, by default, uses **fiorano.jms.ex.sm.def.DefaultJSSESecurityManager** class as a Security Manager. This value can be modified from the Studio using **Profile Manger** at:

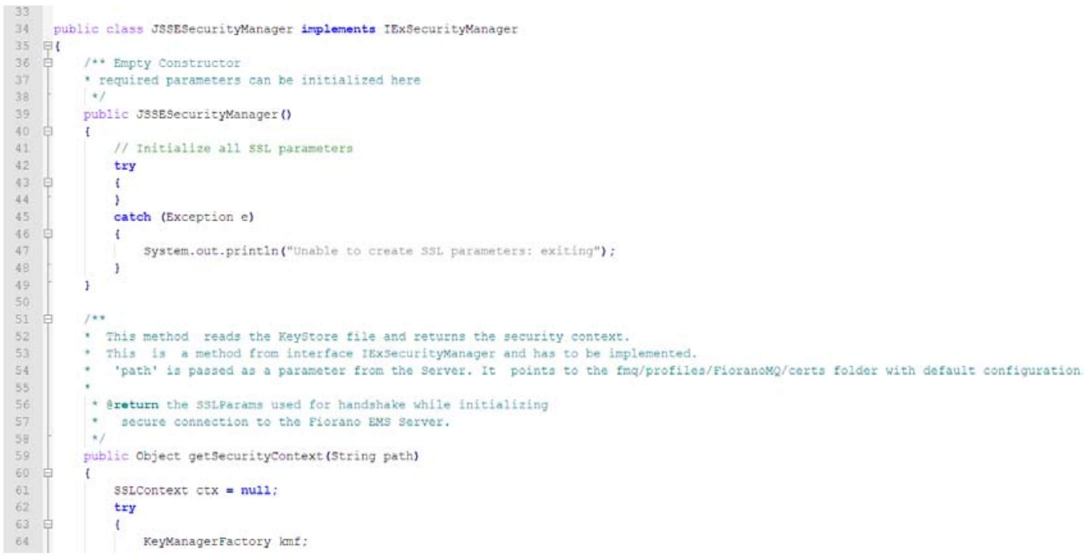
Profiles->FioranoMQ->socketAcceptors->port-1->ConnectionManager[properties]->ManagerClassName

A sample Security Manager Class is displayed below:

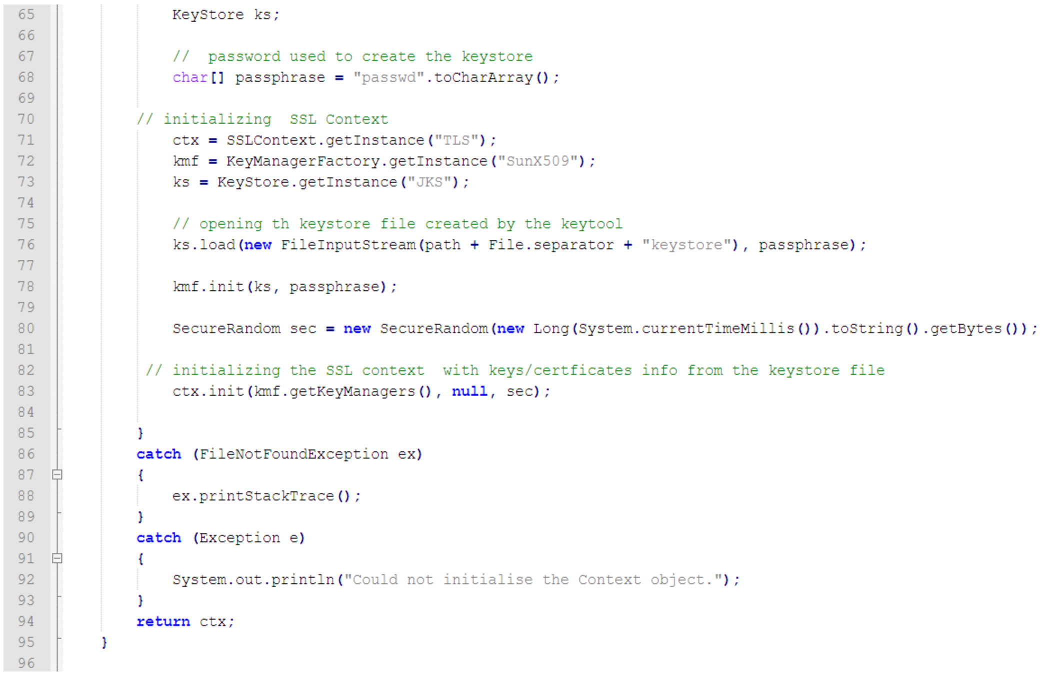
JSSESecurityManager.java



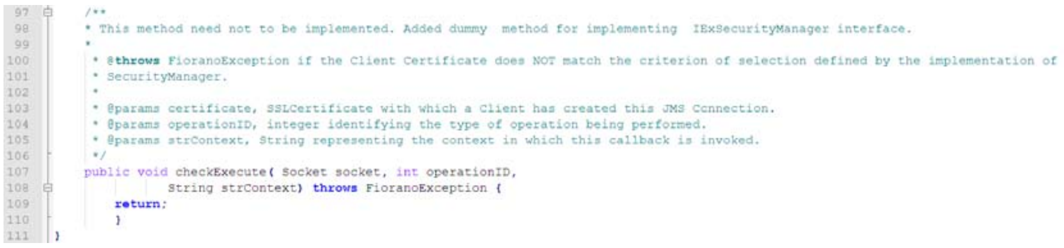
```
/**
 * Copyright (c) 1999-2008, Fiorano Software Technologies Pvt. Ltd.,
 * Copyright (c) 2008-2010, Fiorano Software Pty. Ltd.
 *
 * All rights reserved.
 *
 * This software is the confidential and proprietary information
 * of Fiorano Software ("Confidential Information").  You
 * shall not disclose such Confidential Information and shall use
 * it only in accordance with the terms of the license agreement
 * enclosed with this product or entered into with Fiorano.
 */

import java.io.*;
import java.net.Socket;
import java.security.KeyStore;
import java.security.SecureRandom;
import com.sun.net.ssl.*;

import fiorano.jms.common.FioranoException;
import fiorano.jms.ex.sm.IExSecurityManager;

/**
 * FileName   : JSSESecurityManager.java
 *  This class is a  sample implementation of IExSecurityManager for adding  customer's own SSL implementation to the server.
 * This is used  by the server  when initial SSL handshake process  takes place  while creating a socket.
 *  It  loads digital certificates from appropriate files and initialize the SSLParams.
 *
 * @author FSTPL
 * @created March  5, 2010
 * @version 1.0
 */
```



```java

public class JSSESecurityManager implements IExSecurityManager
{
    /** Empty Constructor
    * required parameters can be initialized here
     */
    public JSSESecurityManager()
    {
        // Initialize all SSL parameters
        try
        {
        }
        catch (Exception e)
        {
            System.out.println("Unable to create SSL parameters: exiting");
        }
    }

    /**
    *  This method  reads the KeyStore file and returns the security context.
    *  This  is  a method from interface IExSecurityManager and has to be implemented.
    *   'path' is passed as a parameter from the Server. It  points to the fmq/profiles/FioranoMQ/certs folder with default configuration
    *
     * @return the SSLParams used for handshake while initializing
     *   secure connection to the Fiorano EMS Server.
     */
    public Object getSecurityContext(String path)
    {
        SSLContext ctx = null;
        try
        {
            KeyManagerFactory kmf;
```



```java
            KeyStore ks;

            //  password used to create the keystore
            char[] passphrase = "passwd".toCharArray();

        // initializing  SSL Context
            ctx = SSLContext.getInstance("TLS");
            kmf = KeyManagerFactory.getInstance("SunX509");
            ks = KeyStore.getInstance("JKS");

            // opening th keystore file created by the keytool
            ks.load(new FileInputStream(path + File.separator + "keystore"), passphrase);

            kmf.init(ks, passphrase);

            SecureRandom sec = new SecureRandom(new Long(System.currentTimeMillis()).toString().getBytes());

         // initializing the SSL context  with keys/certficates info from the keystore file
            ctx.init(kmf.getKeyManagers(), null, sec);

        }
        catch (FileNotFoundException ex)
        {
            ex.printStackTrace();
        }
        catch (Exception e)
        {
            System.out.println("Could not initialise the Context object.");
        }
        return ctx;
    }

```



```java
    /**
    * This method need not to be implemented. Added dummy  method for implementing  IExSecurityManager interface.
    *
     * @throws FioranoException if the Client Certificate does NOT match the criterion of selection defined by the implementation of
     * SecurityManager.
     *
     * @params certificate, SSLCertificate with which a Client has created this JMS Connection.
     * @params operationID, integer identifying the type of operation being performed.
     * @params strContext, String representing the context in which this callback is invoked.
     */
    public void checkExecute( Socket socket, int operationID,
                String strContext) throws FioranoException {
        return;
        }
}
```

4.5.3.1 Compiling the Security Manager

1. compile-client.sh script present in **%Fiorano_Home%\fmq\bin** folder is used to compile the Security Manager class.

2. The following line should be added to **compile-client.conf** (located at the same place as .sh/.bat) file [Under <java.classpath> properties] prior to compilation.

   %FIORANO_HOME%\fmq\lib\server\fmq-sm-api.jar

3. Compile the Security Manager by,

   $> compile-client.sh JSSESecurityManager.java

4.5.3.2 Adding the Security Manager class to the Server's classpath

After compiling the Security Manager class, please add the path under java.classpath [in the fmq.conf (located at %Fiorano_Home%\fmq\bin) file, the class path is specified as:

```
<java.classpath>

../lib

../lib/fmq-kernel.jar

../../extlib/derby/derby.jar

../../licenses

../../xml-catalog
```

The folder containing the **security manager clas**s can be added to this list (or) the **security manager class** can be copied to one of the folders listed.

Enforce Client Authentication. (Refer to the profile screenshot below.): If the **EnforceJSSEAuthentication** parameter is enabled in Connection Manager Configurations then:

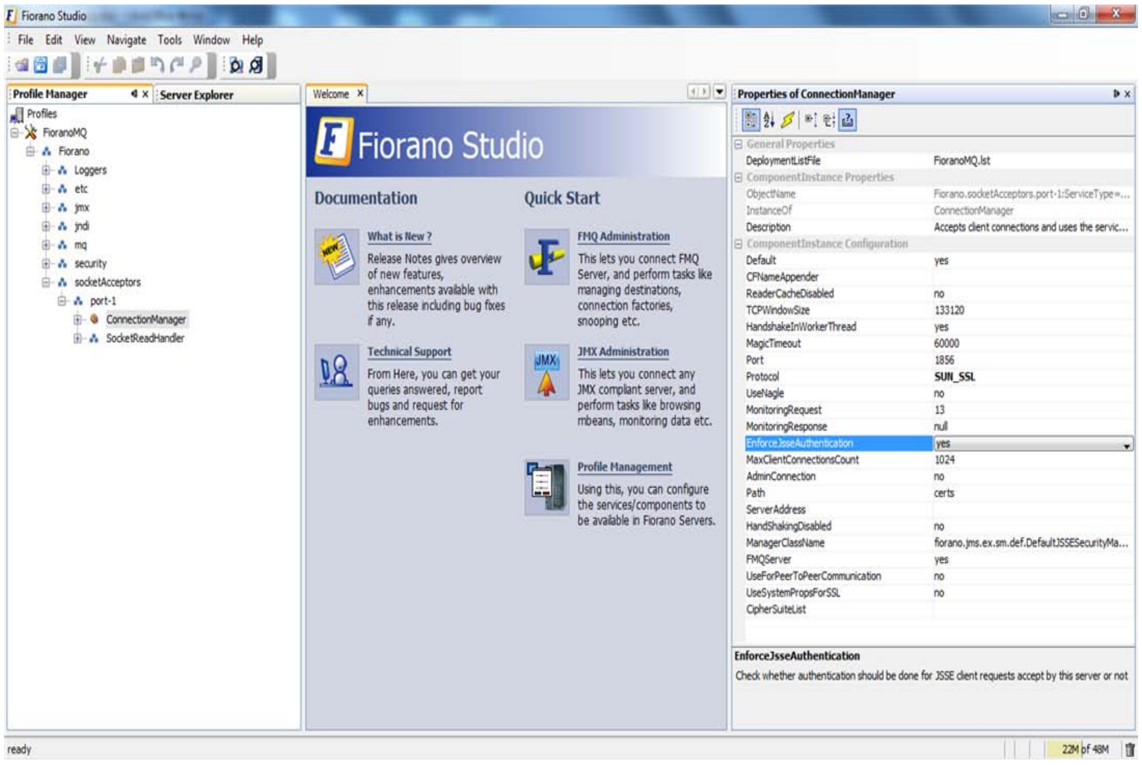
The server validates the certificates provided by the client.(To enable this, the keystore created should be added to the trusted Stores.

The **fmq.conf** file used here.

By default the value is:

javax.net.ssl.trustStore=../profiles/FioranoMQ/certs/jssecacerts. This can be changed to the keystore created,

javax.net.ssl.trustStore=<path - is the path to the keystore>



Properties of ConnectionManager

General Properties	
DeploymentListFile	FioranoMQ.lst
ComponentInstance Properties	
ObjectName	Fiorano.socketAcceptors.port-1:ServiceType=...
InstanceOf	ConnectionManager
Description	Accepts client connections and uses the servic...
ComponentInstance Configuration	
Default	yes
CFNameAppender	
ReaderCacheDisabled	no
TCPWindowSize	133120
HandshakeInWorkerThread	yes
MagicTimeout	60000
Port	1856
Protocol	**SUN_SSL**
UseNagle	no
MonitoringRequest	13
MonitoringResponse	null
EnforceJsseAuthentication	yes
MaxClientConnectionsCount	1024
AdminConnection	no
Path	certs
ServerAddress	
HandShakingDisabled	no
ManagerClassName	fiorano.jms.ex.sm.def.DefaultJSSESecurityMa...
FMQServer	yes
UseForPeerToPeerCommunication	no
UseSystemPropsForSSL	no
CipherSuiteList	

EnforceJsseAuthentication

Check whether authentication should be done for JSSE client requests accept by this server or not

Fiorano Studio
File Edit View Navigate Tools Window Help
Profile Manager
Server Explorer
Profiles
FioranoMQ
Fiorano
Loggers
etc
jmx
jndi
mq
security
socketAcceptors
port-1
ConnectionManager
SocketReadHandler
Welcome
Fiorano Studio
Documentation
Quick Start
What is New ?
Release Notes gives overview of new features, enhancements available with this release including bug fixes if any.
Technical Support
From Here, you can get your queries answered, report bugs and request for enhancements.
FMQ Administration
This lets you connect FMQ Server, and perform tasks like managing destinations, connection factories, snooping etc.
JMX Administration
This lets you connect any JMX compliant server, and perform tasks like browsing mbeans, monitoring data etc.
Profile Management
Using this, you can configure the services/components to be available in Fiorano Servers.
ready
22M of 48M

4.5.4 Client Side Configuration

In order to provide transport layer security on operations involved between the Clients and the FioranoMQ server, the SECURITY_PROTOCOL must be enabled. This can be done by setting the environment variable using the parameter **Context.SECURITY_PROTOCOL** to **FioranoJNDIContextConstants.PROTOCOL_JSSE_SSL**

or **MetaDataConstants.PROTOCOL_JSSE_SSL** before creating InitialContext. For example:

env.put(Context.SECURITY_PROTOCOL, FioranoJNDIContextConstants.PROTOCOL_JSSE_SSL);

(or)

env.put(Context.SECURITY_PROTOCOL, MetaDataConstants.PROTOCOL_JSSE_SSL);

Whenever this variable is set, FMQ's Client library tries to load the Public/Private keys/certificates. This can be done in two different ways – one is to set the System property – **USE_SYSTEM_PROPERTIES_FOR_SSL_TLS**

and provide the system properties for specifying KeyStore and TrustStore locations like :

- javax.net.ssl.keyStore
- javax.net.ssl.keyStorePassword

- javax.net.ssl.keyStoreType (Optional, if not set JDK default will be used)
- javax.net.ssl.trustStore
- javax.net.ssl.trustStorePassword
- javax.net.ssl.trustStoreType (Optional, if not set JDK default will be used)
- javax.net.debug (Optional, if not set JDK default will be used)

Alternatively, an *installed* Client Security Manager can be used to load the required certificates into Client process (described next in this section). In this case, The Security Manager loads the certificates/keys from the Keystore that has been created and the Security manager class should implement **fiorano.jms.runtime.IFMQSecurityManager.** FMQ Client Libraries search for the environment variable set by using **FioranoJNDIContextConstants.SSL_SECURITY_MANAGER** which stores the fully qualified class name of the SecurityManager that implements this interface. For example:

env.put(FioranoJNDIContextConstants.SSL_SECURITY_MANAGER, "com.xxx.yyy.zzz.SomeSecurityManagerImpl");

Please check the Java docs for the class **fiorano.jms.runtime.IFMQSecurityManager** for more information on this API that needs to be implemented.

Certain samples are provided in the fmq/samples/SSLSamples directory for reference on how the class implementing **fiorano.jms.runtime.IFMQSecurityManager** can be used while performing SSL enabled communication with the FioranoMQ Server. Those samples by default use JSSESecurityManager class (also provided along with the samples) as a Security Manager. The fully qualified class name is passed as an environment variable

FioranoJNDIContext.SSL_SECURITY_MANAGER

while creating the InitialContext from the client. A sample Security Manager Class is shown below for your convenience.

JSSESecurityManager.java

```
/**

 * Copyright © 2008-2010, Fiorano Software Pte. Ltd. and affiliates.

 *

 * All rights reserved.

 *

 * This software is the confidential and proprietary information

 * of Fiorano Software ("Confidential Information").  You

 * shall not disclose such Confidential Information and shall use
 * it only in accordance with the terms of the license agreement

 * enclosed with this product or entered into with Fiorano.

 */




import fiorano.jms.common.FioranoException;

import fiorano.jms.runtime.IFMQSecurityManager;


import javax.net.ssl.KeyManagerFactory;

import javax.net.ssl.SSLContext;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.net.Socket;

import java.security.KeyStore;

import java.security.SecureRandom;


/**

 * Install a server certificate handler callback which is

 * invoked at the time of creating connection to server.

 * This is used for server authentication by Client.

 *

 * @author FSIPL

 * @version 1.0

 * @created December 31, 2007

 */

public class JSSESecurityManager implements IFMQSecurityManager {
    //  Set this path to your installation of FMQ_DIR

    //

    static String FMQ_INSTALL_DIR = System.getProperty("FMQ_DIR");

    static String FMQ_CERTS_DIR = System.getProperty("CERTS_DIR");



    // This lookup Dir assumes that default profile from whcih certs should be

    // fetched is "FioranoMQ"

    static String lookUpDir = FMQ_INSTALL_DIR + File.separator + "profiles" + File.separator +
"FioranoMQ" +

            File.separator + "certs" + File.separator;




    /**

     */

    public JSSESecurityManager() {

        // Initialize all SSL parameters

        try {

        }

        catch (Exception e) {

            System.out.println("Unable to create SSL parameters: exiting");

        }

    }


    /**

     * @return the SSLParams used for handshake while initializing

     *         secure connection to the Fiorano EMS Server.

     */

    public Object getSecurityContext() {
        SSLContext ctx = null;


        try {

            KeyManagerFactory kmf;

            KeyStore ks;

            char[] passphrase = "passphrase".toCharArray();


            ctx = SSLContext.getInstance("TLS");

            kmf = KeyManagerFactory.getInstance("SunX509");

            ks = KeyStore.getInstance("JKS");


            String certsDir = FMQ_CERTS_DIR;


            certsDir = (certsDir == null) ? lookUpDir : certsDir + File.separator;


            ks.load(new FileInputStream(certsDir + "DefaultJSSEKeys"), passphrase);

            kmf.init(ks, passphrase);


            SecureRandom sec = new SecureRandom(new
Long(System.currentTimeMillis()).toString().getBytes());


            ctx.init(kmf.getKeyManagers(), null, sec);


        }

        catch (FileNotFoundException ex) {

            ex.printStackTrace();

        }

        catch (Exception e) {
           System.out.println("Could not initialise the Context object.");

        }

        return ctx;

     }



     /**

      * @param socket

      * @throws FioranoException

      * @throws JMSException     if the certificate received is not

      *                          from a valid server.

      */

     public void checkExecute(Socket socket)

           throws FioranoException {

        return;

     }

}
```

4.5.4.1 Compiling the Security Manager

1. compile-client.sh script present in %Fiorano_Home%\fmq\bin folder is used to compile the Security Manager class.
2. The following line should be added to compile-client.conf (located at the same place as .sh/.bat) file [Under <java.classpath> properties] before compiling.

   %FIORANO_HOME%\fmq\lib\client\fmq-common-api.jar
3. 3. Compile the Security Manager by, $> compile-client.sh JSSESecurityManager.java

4.5.4.2 Adding the Security Manager class to the Client's classpath

After compiling the Security Manager class, please add the path under java.classpath. In run-client.conf (this file is also located at %Fiorano_Home%\fmq\bin) file, the class path is specified as following:

```
<java.classpath>

%FMQ_DIR%/lib/fmq-rtl.jar
```

%FMQ_DIR%/samples/jndiProperties

%JAVA_HOME%/lib/jndi.jar

%JAVA_HOME%/lib/tools.jar

%JAVA_HOME%/lib/classes.zip

# If you are using TibEms bridge

#path-to-TibEMS/clients/java/tibjms.jar

Either the folder or jar that contains security manager class can be added to this list.

Important: From FioranoMQ 9.3.0 release onwards, the classes which are used for initializing Key and Trust managers while loading the keys from the KeyStore have been changed from com.sun.net.ssl.* package to javax.net.ssl.* package. This is as per the changes involved in the Sun's JDK 5 as the APIs provided in com.sun.net.ssl.* package have been deprecated. Henceforth, FioranoMQ only supports the usage of javax.net.ssl.* package's classes. These classes include:

com.sun.net.ssl.SSLContext → javax.net.ssl.SSLContext

com.sun.net.ssl.KeyManager → javax.net.ssl.KeyManager

com.sun.net.ssl.KeyManagerFactory → javax.net.ssl.KeyManagerFactory

com.sun.net.ssl..TrustManager → javax.net.ssl.TrustManager

com.sun.net.ssl.TrustManagerFactory → javax.net.ssl.TrustManagerFactory

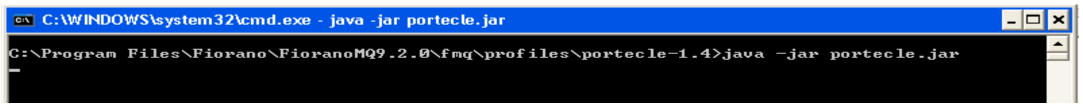
4.5.5 Creating Certificates for OpenSSL in C++

.pem format or converting the above generated keystore to PEM encoding

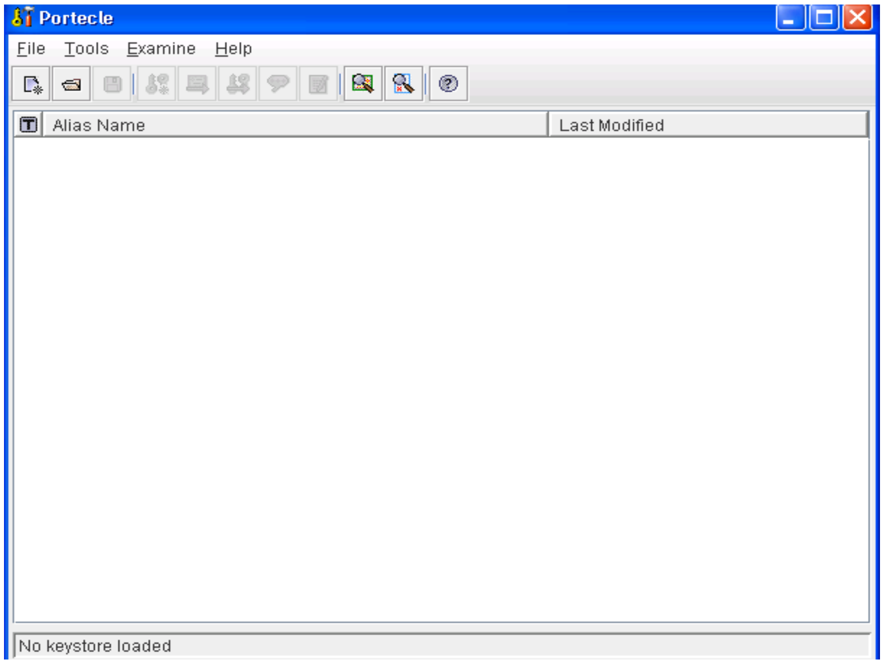
Portcele is a java tool with an UI for managing the Java certificates. It can be downloaded at http://sourceforge.net/projects/portecle/. This tool is used to convert java certificates into other compatible formats.

To start this tool, type in the command,

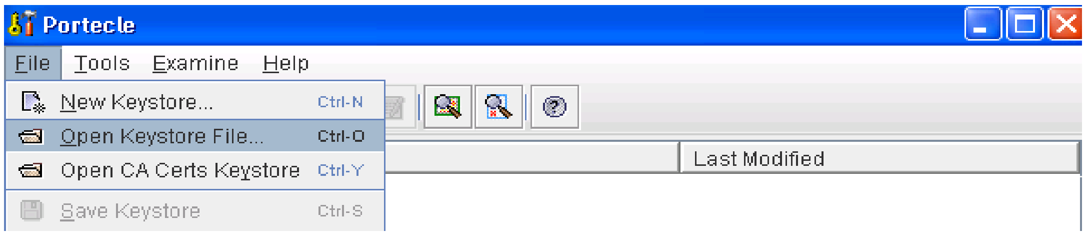
java -jar portecle.jar

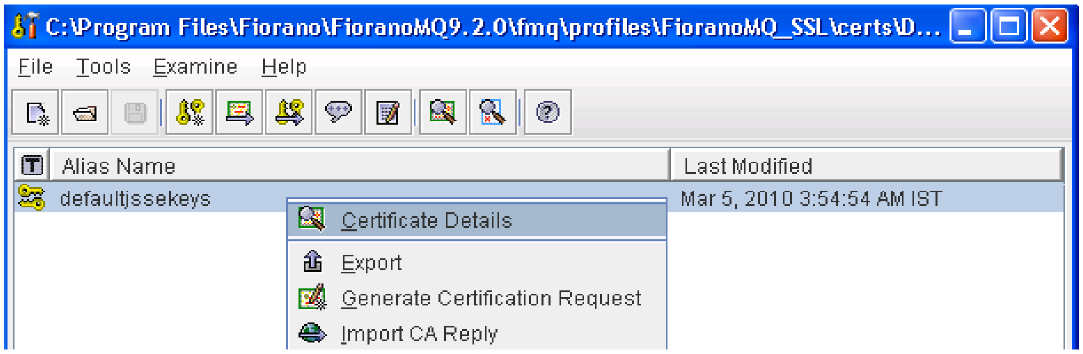


```
C:\Program Files\Fiorano\FioranoMQ9.2.0\fmq\profiles\portecle-1.4>java -jar portecle.jar
_
```



1. Open the keystore that needs converting to the .PEM format using Portcele. All the information related to the certificate can be viewed here.
2. Navigate to **File >> Open KeyStore File.** Choose the keystore that was created.
3. Enter the password provided at the time of the creation of the keystore.




C:\Program Files\Fiorano\FioranoMQ9.2.0\fmq\profiles\FioranoMQ_SSL\certs\D...
File Tools Examine Help
Alias Name
Last Modified
defaultjssekeys
Mar 5, 2010 3:54:54 AM IST
Certificate Details
Export
Generate Certification Request
Import CA Reply

4. Right-click on **certificate** and choose **export**. Select the export 'type' as "**Private Key and Certificate**s" and the export format as "**PEM Encoding**".
5. Enter the **password** after clicking '**OK**'.
6. Enter the **password** that was used to create the keystore using the java keytool.
7. Save the file as **clientCert.pem** or any other name.

This file is used by the C++ RTL for SSL communication.

The client side configurations in C++ lies within the environment variables used in creating the initial context as shown below:

```
m_env->Put(SSL_CERT_FILE,certfile);

m_env->Put(SSL_PRIVATE_KEY_FILE,keyfile);

m_env->Put(SSL_PRIVATE_KEY_PASSWORD,newMqString("passwd"));
```

In the code above, certfile and keyfile refer to the location of the clientCert.pem created using portcele.

By default, the C++ samples use the 'dsa-client-cert.pem' for the certificate and 'enc-dsa-client-key.pem' for the private key.

'passwd', is the password that is used when creating the certificate.

Default certificates are located at %Fiorano_Home%\fmq\clients\c\native\certs.

4.6 Looking up

4.6.1 JNDI Environment

The following parameters should be specified within the environment passed to JNDI when looking up an admin object from the FioranoMQ Server. The FioranoMQ Server would be running on the default socket acceptor configuration.

```
// Modify the IP, Port according to the server's socket acceptor configuration

String url = “http://localhost:1856”;



// environment passed to jndi

Hashtable env = new Hashtable ( );

env.put (Context.SECURITY_PRINCIPAL, “anonymous”);

env.put (Context.SECURITY_CREDENTIALS, “anonymous”);

env.put (Context.PROVIDER_URL, url);

env.put(Context.INITIAL_CONTEXT_FACTORY,”fiorano.jms.runtime.naming.FioranoInitialContextFactory”);
```

Note:

- An application can use the actual string values of the static variables defined in javax.naming.Context class.

- An application can also specify these parameters in a file called jndi.properties, located in the working directory of the application. The user code does not need to pass any parameters programmatically.

4.6.2 Looking up from Server Running on HTTP Protocol

In addition to specifying the jndi parameters, as mentioned in the preceding section, the application needs to specify the protocol as HTTP.

This is done by adding the following line to the application code:

```
env.put(FioranoJNDIContext.TRANSPORT_PROTOCOL, FioranoJNDIContext.PROTOCOL_HTTP);
```

Note: Since the code uses the FioranoJNDIContext class, the following must be added to the code:

```
import fiorano.jms.runtime.naming.FioranoJNDIContext;
```

Code modifications are not required when viewed from an external JNDI repository or when picking up environment variables from jndi.properties.

4.6.3 Viewing from Server Running on JSSE Protocol

When viewing from a server running on JSSE protocol, the application, besides specifying the jndi parameters specified above, should also specify the protocol as JSSE_SSL. In addition, it should specify the security manager. This can be done by adding the following line to the application code:

```
env.put (Context.SECURITY_PROTOCOL, FioranoJNDIContext.PROTOCOL_JSSE_SSL);

env.put (FioranoJNDIContext.SSL_SECURITY_MANAGER, "JSSESecurityManager");
```

Note: Since the code uses the FioranoJNDIContext class, the following must be added to the code:

```
import fiorano.jms.runtime.naming.FioranoJNDIContext;
```

4.6.4 Looking up from Server Running on LPC Protocol

When viewing from a server running on LPC protocol, the application should specify the protocol as LPC in addition to specifying the jndi parameters mentioned above.

This is done by adding the following line to the application code:

```
env.put(FioranoJNDIContext.TRANSPORT_PROTOCOL, FioranoJNDIContext.PROTOCOL_LPC);
```

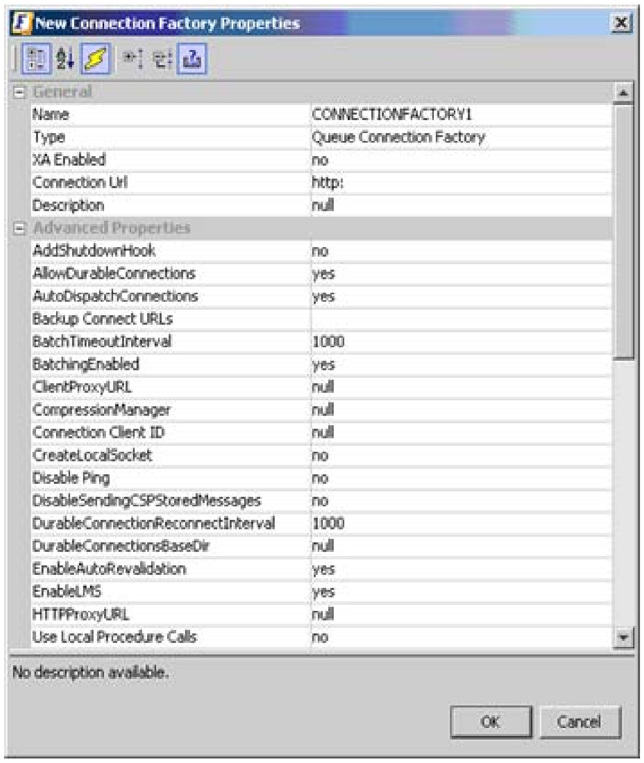
Note: Since the code uses the FioranoJNDIContext class, the following must be added to the code:

```
import fiorano.jms.runtime.naming.FioranoJNDIContext;
```

4.7 Connection Factory

4.7.1 Creating a Connection Factory

1. Launch the Fiorano Studio and click on the **Server Explore** pane. Select the desired server and right-click. Select **Login** from the pop-up menu.
2. Right-click on the **Connection Factories** node in the selected tree and select **Add Connection Factory** from the pop-up menu. Specify the connection factory Name, Connection URL and other parameters as desired, and click the **OK** button.



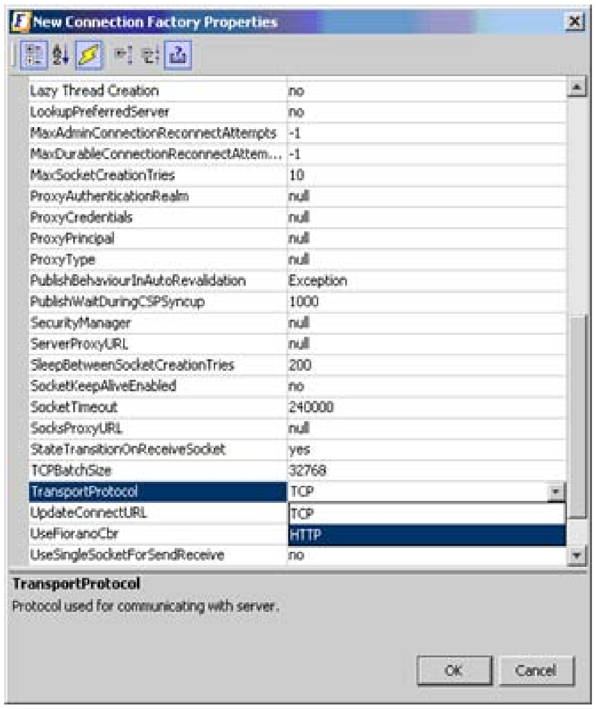
General	
Name	CONNECTIONFACTORY1
Type	Queue Connection Factory
XA Enabled	no
Connection Url	http:
Description	null
Advanced Properties	
AddShutdownHook	no
AllowDurableConnections	yes
AutoDispatchConnections	yes
Backup Connect URLs	
BatchTimeoutInterval	1000
BatchingEnabled	yes
ClientProxyURL	null
CompressionManager	null
Connection Client ID	null
CreateLocalSocket	no
Disable Ping	no
DisableSendingCSPStoredMessages	no
DurableConnectionReconnectInterval	1000
DurableConnectionsBaseDir	null
EnableAutoRevalidation	yes
EnableLMS	yes
HTTPProxyURL	null
Use Local Procedure Calls	no

No description available.

Note: The default connection factory parameters are configured for the default socket acceptor settings (TCP, non SSL).

4.7.2 Creating an HTTP Enabled Connection Factory

1. Create a connection factory as explained in Section 4.7.1.
2. Modify the Protocol to **HTTP** and click the **OK** button.



Lazy Thread Creation	no
LookupPreferredServer	no
MaxAdminConnectionReconnectAttempts	-1
MaxDurableConnectionReconnectAttem...	-1
MaxSocketCreationTries	10
ProxyAuthenticationRealm	null
ProxyCredentials	null
ProxyPrincipal	null
ProxyType	null
PublishBehaviourInAutoRevalidation	Exception
PublishWaitDuringCSPSyncup	1000
SecurityManager	null
ServerProxyURL	null
SleepBetweenSocketCreationTries	200
SocketKeepAliveEnabled	no
SocketTimeout	240000
SocksProxyURL	null
StateTransitionOnReceiveSocket	yes
TCPBatchSize	32768
TransportProtocol	TCP
UpdateConnectURL	TCP
UseFioranoCbr	HTTP
UseSingleSocketForSendReceive	no

TransportProtocol
Protocol used for communicating with server.

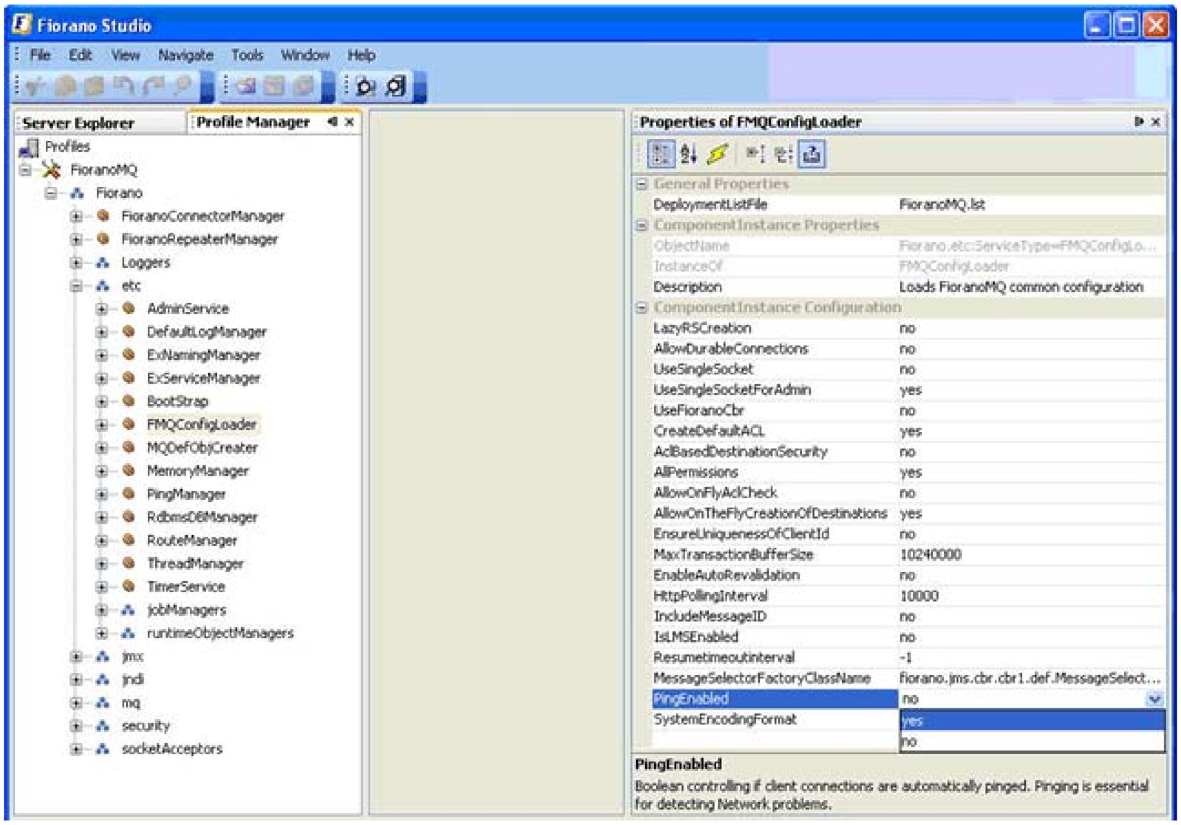
4.8 Pinging

4.8.1 How to Enable Pinging

Note: This configuration is done in the offline mode

1. Launch the Fiorano Studio and open the **Profile Manager**. Right-click the **Profiles** node and select **Open Profile** from the pop-up menu. Select the desired profile and click the **Open** button.
2. Navigate to **FMQConfigLoader** MBean. It can be found in domain **Fiorano -> etc**.

3. In the properties panel set the parameter **PingEnabled** to ‘**yes’**.



General Properties	
DeploymentListFile	FioranoMQ.lst
ComponentInstance Properties	
ObjectName	Fiorano.etc:ServiceType=FMQConfigLo...
InstanceOf	FMQConfigLoader
Description	Loads FioranoMQ common configuration
ComponentInstance Configuration	
LazyRSCreation	no
AllowDurableConnections	no
UseSingleSocket	no
UseSingleSocketForAdmin	yes
UseFioranoCbr	no
CreateDefaultACL	yes
AclBasedDestinationSecurity	no
AllPermissions	yes
AllowOnFlyAclCheck	no
AllowOnTheFlyCreationOfDestinations	yes
EnsureUniquenessOfClientId	no
MaxTransactionBufferSize	10240000
EnableAutoRevalidation	no
HttpPollingInterval	10000
IncludeMessageID	no
IsLMSEnabled	no
ResumetimeoutInterval	-1
MessageSelectorFactoryClassName	fiorano.jms.cbr.cbr1.def.MessageSelect...
PingEnabled	no
SystemEncodingFormat	yes
	no

PingEnabled

Boolean controlling if client connections are automatically pinged. Pinging is essential for detecting Network problems.

4. Save the profile.

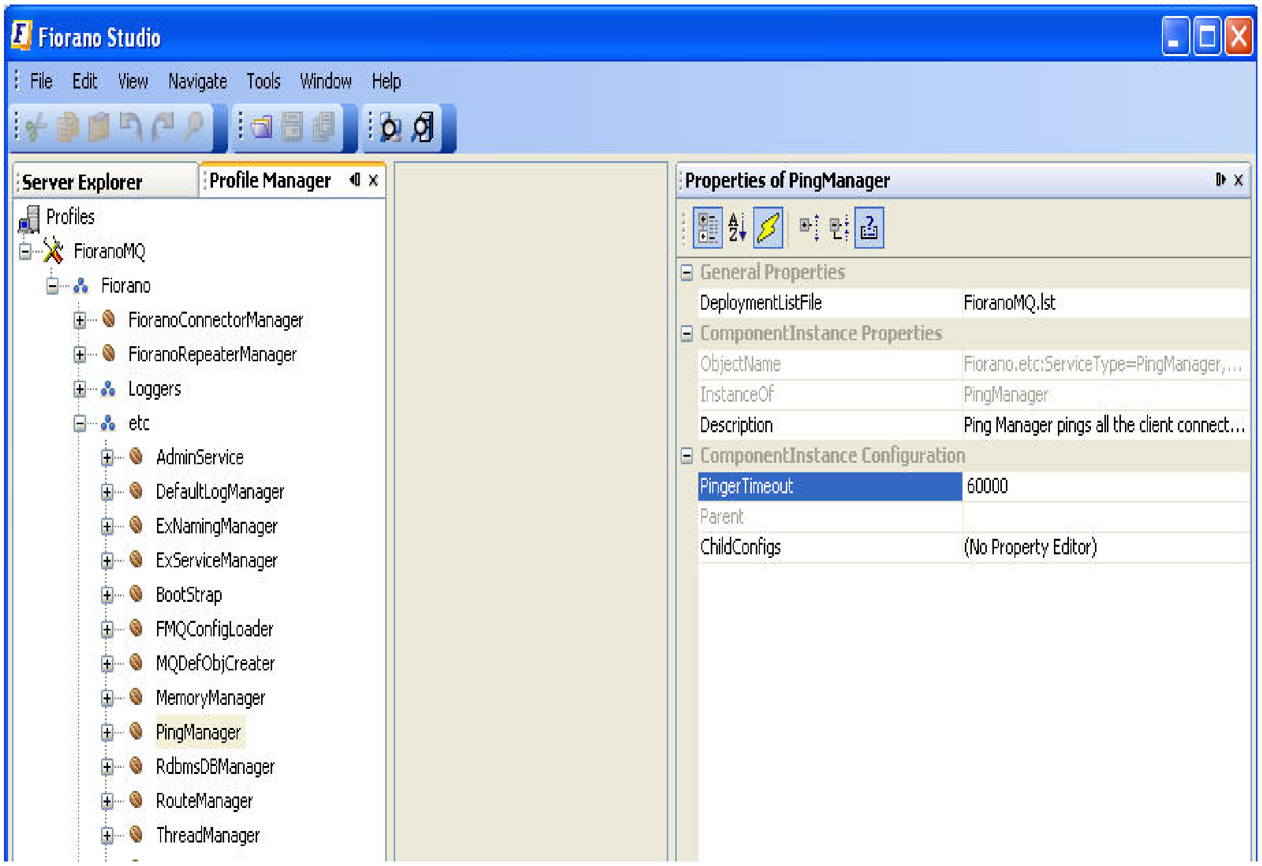
Note: From FioranoMQ 9.1.0 release onwards, the PingEnabled parameter is set to '**true'** as its default value. This property can be controlled if client connections are automatically pinged. Pinging is essential for detecting Network problems. Enabling ‘pinging’ will make sure that connections that are no longer valid are properly closed and the resources they use are cleaned up in the FioranoMQ Server.

4.8.2 Modifying Ping Timeout Interval

Note: This configuration is done in offline mode.

1. Launch the Fiorano Studio. Open the **Profile Manager** and open the desired profile as explained in section 4.8.1 How to Enable Pinging.
2. Navigate to **PingManager** MBean. It can be found in domain **Fiorano -> etc**.

3. Modify the parameter **Pinger Timeout** as shown in the properties panel of **Mbean** to the desired value (in milliseconds).


Fiorano Studio
File Edit View Navigate Tools Window Help
Server Explorer
Profile Manager
Properties of PingManager

Profiles
- FioranoMQ
  - Fiorano
    - FioranoConnectorManager
    - FioranoRepeaterManager
    - Loggers
    - etc
      - AdminService
      - DefaultLogManager
      - ExNamingManager
      - ExServiceManager
      - BootStrap
      - FMQConfigLoader
      - MQDefObjCreater
      - MemoryManager
      - PingManager
      - RdbmsDBManager
      - RouteManager
      - ThreadManager

General Properties	
DeploymentListFile	FioranoMQ.lst
ComponentInstance Properties	
ObjectName	Fiorano.etc:ServiceType=PingManager,...
InstanceOf	PingManager
Description	Ping Manager pings all the client connect...
ComponentInstance Configuration	
PingerTimeout	60000
Parent	
ChildConfigs	(No Property Editor)

4. Right-click on the FioranoMQ node and select **Save** option from the pop-up menu.

4.8.3 Verifying Ping Setup

Ping can be verified by following any one of the approaches mentioned below:

Increase the trace level for Logger named **Fiorano -> FMQ -> Ping**. In the **Properties of Ping** pane change the **LogLevel** to **Verbose**. This results in generating ping related logs in the server.

Using an application, create a connection. Set an exception listener and start the connection. Disable the network connection. Exception listener's onException should be carried out within the ping timeout interval.

4.9 FioranoMQ HTTP Support

4.9.1 Using HTTP with FioranoMQ

For more details about Using HTTP with FioranoMQ, please refer to section 23.2.1 Adding a Connection Factory.

4.10 Client Side Requirements

While switching the protocol in the server from TCP to HTTP, the following changes must be made at the application level:

- Pass additional parameters as JNDI environment if viewed from FioranoMQ. These parameters are described in section 23.1.2 Over HTTP Protocol.
- Use an HTTP Enabled Connection factory. Instructions for creating an HTTP Enabled connection factory can be found in section 4.7.2 Creating an HTTP Enabled Connection Factory
- Include HTTPClient.zip in the classpath if not already included.

4.11 Using Proxies

Connecting an application to the HTTP Proxy Server 192.168.100.37 on port 8080 can be done in the following ways:

(a) By setting the host and port as parameters in client applications:

```
env.put (FioranoJNDIContext.HTTP_PROXY_URL, "http://192.168.100.37:8080");
```

If the client is to connect to the SOCKS proxy Server

```
env.put (FioranoJNDIContext.SOCKS_PROXY_URL, "http://192.168.100.37:1080");
```

(b) By setting JVM parameters through the run-client.bat (run-client.sh for UNIX Systems) file:

Modify the run-client.bat (run-client.sh on UNIX Systems) so as to add the following arguments to the VM (VM properties):

```
-Dhttp.proxyHost=192.168.100.37 -Dhttp.proxyPort=8080
```

If the client is to connect to the SOCKS proxy Server 192.168.100.37 on port 1080, modify run-client.bat (run-client.sh on UNIX Systems):

```
-Dhttp.socksHost=192.168.100.37 -Dhttp.socksPort=1080
```

Client applications can be customized for popular proxy servers such as MicrosoftISAProxy and Netscape Proxy, using the HTTP_PROXY_TYPE parameter. This parameter can be specified in client applications:

```
env.put (FioranoJNDIContext.HTTP_PROXY_TYPE,FioranoJNDIContext.MS_ISA_PROXY);
```

4.11.1 Proxy Authentication

Various Proxy Authentication parameters such as the Authentication Realm username and password can be specified from the client application as JNDi environment variables:

```
env.put (FioranoJNDIContext.PROXY_AUTHENTICATION_REALM, "LDAP");
env.put (FioranoJNDIContext.PROXY_PRINCIPAL, "fiorano");

env.put (FioranoJNDIContext.PROXY_CREDENTIALS, "fiorano");
```

4.12 Tunneling Through Firewalls

Consider a scenario where client applications are protected by a corporate firewall and need to use the services of FioranoMQ server through SOCKS tunneling. The following code illustrates how the clients' applications, even when protected by firewalls, can access the services of the FioranoMQ server by tunneling through client side firewalls.

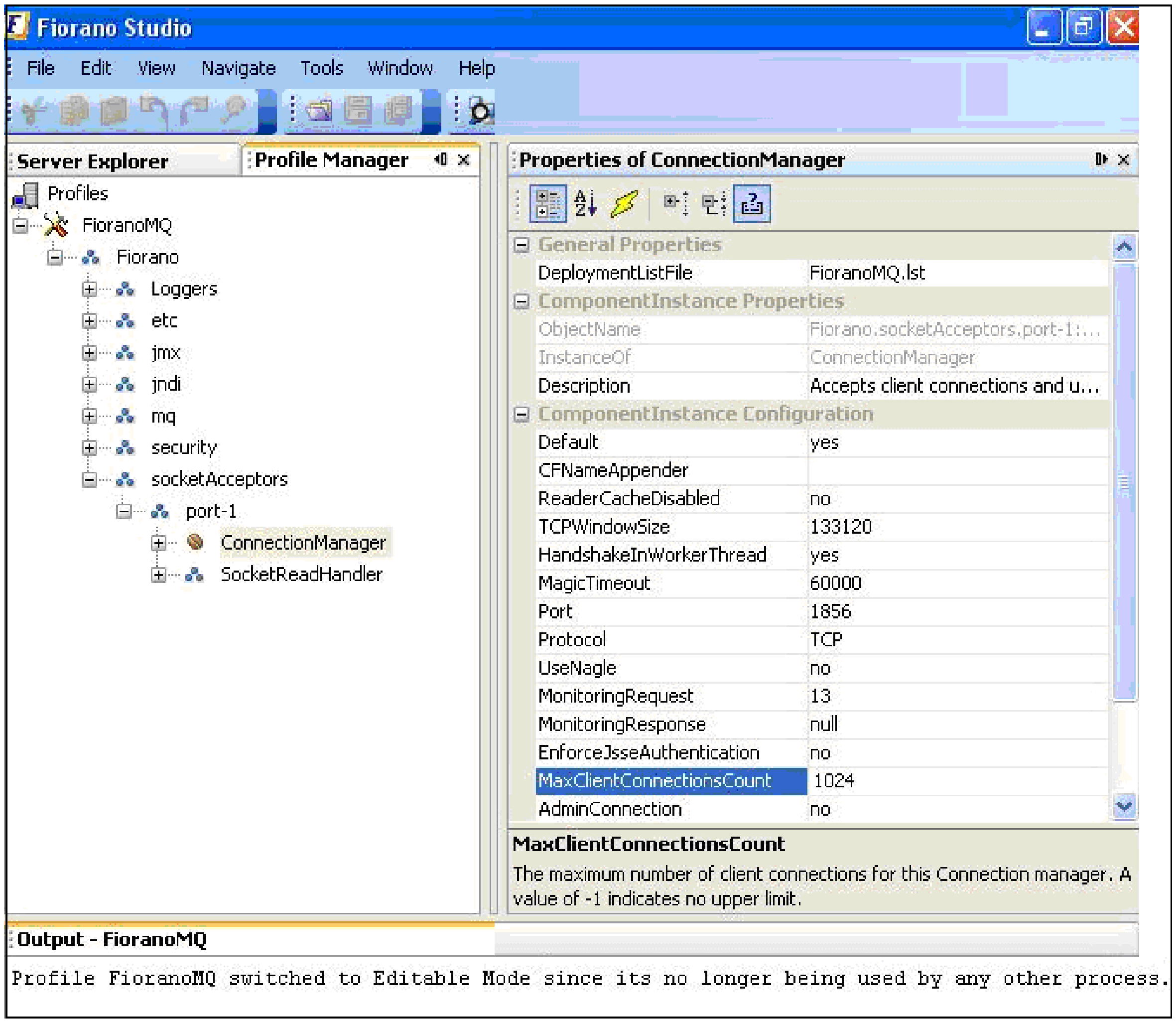
```
//  This code fragment expects the args[] to contain

// clientproxyName, clientProxyPort, FioranoMQ 9ServerAddress,

// FioranoMQ 9Server Port.



public void sendData(String[] args)

{

try

{

//Initialize firewall Settings

String proxyName = args[0];

int proxyPort = Integer.parseInt (args[1]);

//Initialize FioranoMQ 9 Server Settings

String serverName = args[2];

int serverPort = Integer.parseInt (args[3]);



// 1. Create the InitialContext Object used for

// looking up JMS administered objects

// Set the Client Proxy Address/port,

// The 1st argument is set to NULL to indicate

// that there is no security parameter that has

// been set. This parameter is set for
// SSL Tunneling

FioranoInitialContext ic = new FioranoInitialContext ();

// Set System property to indicate proxyHost and
// proxy Port. All calls now get routed through
// the SOCKS Server

Properties property = System.getProperties();
property.put ("socksProxyPort",""+proxyPort);
property.put ("socksProxyHost",proxyName);
System.setProperties (property);

// Bind the InitialContext to Server
ic.bind (InetAddress.getByName(serverName),
serverPort);

// Lookup Connection Factory and Topic names
TopicConnectionFactory tcf =(TopicConnectionFactory) ic.lookup("primaryTCF");
Topic topic = (Topic)ic.lookup("primaryTopic");

// 4.2 Dispose the InitialContext resources
//
ic.dispose();

// 2. Create and start a topic connection
System.out.println("Creating topic connection");
```

```
TopicConnection tc = tcf.createTopicConnection();

tc.start ();


// 3. Create a topic session on this connection

TopicSession ts = tc.createTopicSession(false,1);


// 4. Create a publisher for this topic

TopicPublisher tp = ts.createPublisher(topic);

System.out.println ("Ready to publish messages :

Enter Q to Quit...");


// 5. Create a text message for use in the ‘while’

// loop

TextMessage textmsg1 = ts.createTextMessage();


// 6. Read in data from standard input and publish

// it in a loop

while (true)

{

BufferedReader br = new BufferedReader

(new InputStreamReader(System.in), 1);

System.out.print("Enter a Message to be

published : ");

String str = br.readLine();

// Set and Publish the message

textmsg4.setText(str);

tp.publish(textmsg1);
// Break out of this loop when done

if (str.equalsIgnoreCase ("Q") )

break;

}

System.out.println("Closing topic session and topic connection");

ts.close();

tc.close();

}

catch(Exception ex)

{

    ex.printStackTrace();

}
```

4.13 Configure Maximum Client Connections

To configure maximum client connections do the following:

1. Open the desired profile for off-line editing through the **Profile Manager** in Studio as explained in section 4.8.1 Navigate to the Profile Manager. Select **Fiorano->socketAcceptors->port1->ConnectionManager** as shown in the figure below:



General Properties	
DeploymentListFile	FioranoMQ.lst
ComponentInstance Properties	
ObjectName	Fiorano.socketAcceptors.port-1:...
InstanceOf	ConnectionManager
Description	Accepts client connections and u...
ComponentInstance Configuration	
Default	yes
CFNameAppender	
ReaderCacheDisabled	no
TCPWindowSize	133120
HandshakeInWorkerThread	yes
MagicTimeout	60000
Port	1856
Protocol	TCP
UseNagle	no
MonitoringRequest	13
MonitoringResponse	null
EnforceJsseAuthentication	no
MaxClientConnectionsCount	1024
AdminConnection	no

MaxClientConnectionsCount
The maximum number of client connections for this Connection manager. A value of -1 indicates no upper limit.

```
Profile FioranoMQ switched to Editable Mode since its no longer being used by any other process.
```

2. In the Properties of **ConnectionManager** pane, change the property of **MaxClientConnectionsCount** to the value desired.

3. Right-click on the **FioranoMQ** node and select **Save** from the pop-up menu.

Note: The same can be configured through JMX at **FMQ-JMX Connection->Fiorano->socketAcceptors->port-1->ConnectionManager->ConnectionManager->config**.

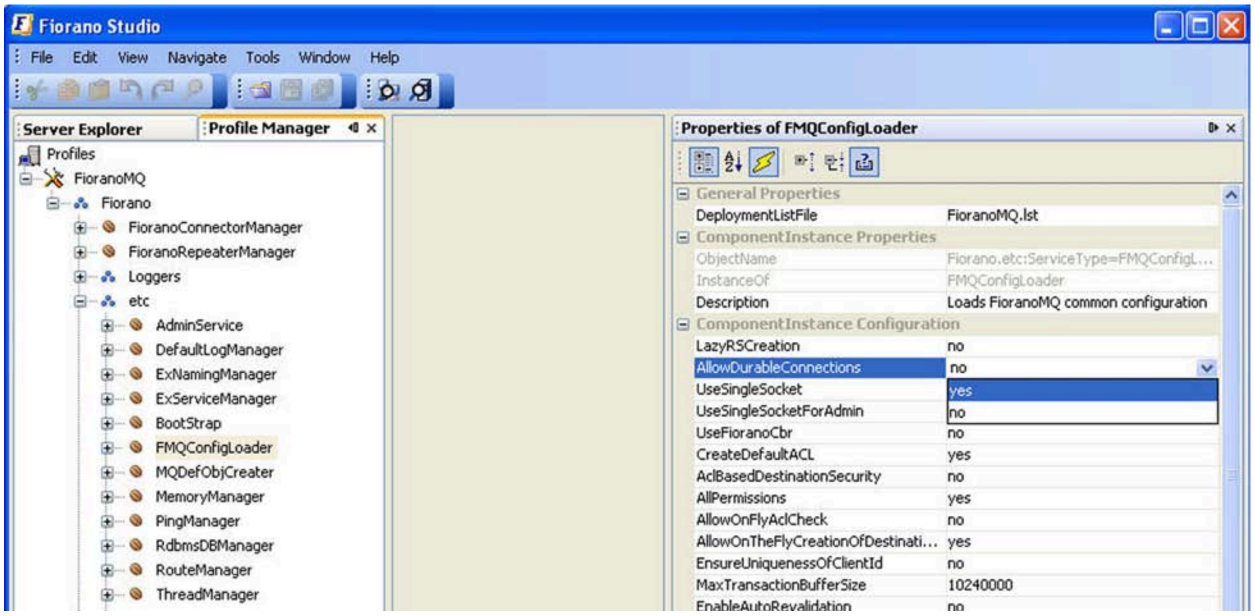
Chapter 5: Durable Connections

For information about configuring profiles through a text-based file, see 'Getting Started' in FioranoMQ.

5.1 Durable Connections in the Server

The following steps enable durable connections in the server:

1. Launch the **admin studio** and open the **profile** (which by default is FioranoMQ) in the offline mode.
2. Navigate to **FioranoMQ -> Fiorano -> etc -> FMQConfigLoader**. In the properties pane, set the **AllowDurableConnections** property to '**yes'**.
3. **Save** the configuration.



General Properties	
DeploymentListFile	FioranoMQ.lst
ComponentInstance Properties	
ObjectName	Fiorano.etc:ServiceType=FMQConfigL...
InstanceOf	FMQConfigLoader
Description	Loads FioranoMQ common configuration
ComponentInstance Configuration	
LazyRSCreation	no
AllowDurableConnections	no
UseSingleSocket	yes
UseSingleSocketForAdmin	no
UseFioranoCbr	no
CreateDefaultACL	yes
AclBasedDestinationSecurity	no
AllPermissions	yes
AllowOnFlyAclCheck	no
AllowOnTheFlyCreationOfDestinati...	yes
EnsureUniquenessOfClientId	no
MaxTransactionBufferSize	10240000
EnableAutoRevalidation	no

5.1.1 Enabling Durable Connections for a Client Application

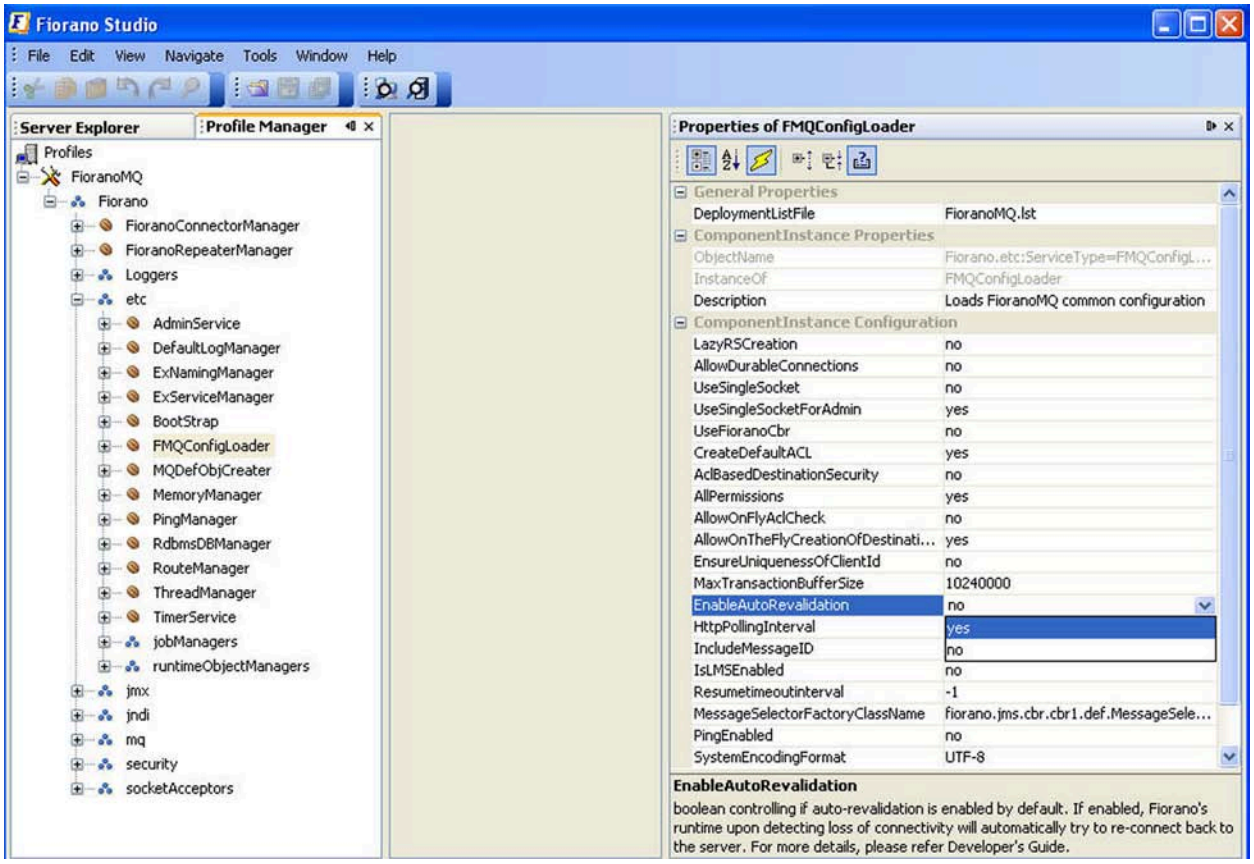
```
// Create the InitialContext Object used for looking up

//    JMS administered objects on the Fiorano/EMS

//    located on the default host.

//

Hashtable env = new Hashtable();
```

```
env.put(Context.SECURITY_PRINCIPAL, "anonymous");

env.put(Context.SECURITY_CREDENTIALS, "anonymous");

env.put(Context.PROVIDER_URL, m_url);

env.put(Context.INITIAL_CONTEXT_FACTORY,
"fiorano.jms.runtime.naming.FioranoInitialContextFactory");

env.put(FioranoJNDIContext.ALLOW_DURABLE_CONNECTIONS, "true");



InitialContext ic = new InitialContext(env);

System.out.println("Created InitialContext :: " + ic);
```

5.2 Auto Revalidation

Follow the steps mentioned below to enable auto-revalidation:

1. Launch the **admin studio** and open the **profile** (which by default is FioranoMQ) in the offline mode.
2. Navigate to **FioranoMQ -> Fiorano -> etc -> FMQConfigLoader**. In the properties pane, set the **EnableAutoRevaildation** property to '**yes'**.
3. **Save** the configuration.



Properties of FMQConfigLoader

General Properties	
DeploymentListFile	FioranoMQ.lst
ComponentInstance Properties	
ObjectName	Fiorano.etc:ServiceType=FMQConfigL...
InstanceOf	FMQConfigLoader
Description	Loads FioranoMQ common configuration
ComponentInstance Configuration	
LazyRSCreation	no
AllowDurableConnections	no
UseSingleSocket	no
UseSingleSocketForAdmin	yes
UseFioranoCbr	no
CreateDefaultACL	yes
AclBasedDestinationSecurity	no
AllPermissions	yes
AllowOnFlyAclCheck	no
AllowOnTheFlyCreationOfDestinati...	yes
EnsureUniquenessOfClientId	no
MaxTransactionBufferSize	10240000
EnableAutoRevalidation	no
HttpPollingInterval	yes
IncludeMessageID	no
IsLMSEnabled	no
Resumetimeoutinterval	-1
MessageSelectorFactoryClassName	fiorano.jms.cbr.cbr1.def.MessageSele...
PingEnabled	no
SystemEncodingFormat	UTF-8

EnableAutoRevalidation

boolean controlling if auto-revalidation is enabled by default. If enabled, Fiorano's runtime upon detecting loss of connectivity will automatically try to re-connect back to the server. For more details, please refer Developer's Guide.

Note: Auto-revalidation is turned on automatically if Durable Connections are enabled. If auto-revalidation-enabled or durable connection is disconnected by the server (using WMT or Studio), the disconnection will not persist for a long duration. This is because, by definition, properties (EnableAutoRevalidation, AllowDurableConnections) dictate that the client should re-establish connection with the server. Therefore, a connection can only be disconnected when a client closes the connection.

5.2.1 Enabling Auto-Revalidation for a Client Application

```
// 1. Create the InitialContext Object used for looking up

//    JMS administered objects on the FioranoMQ 9

//    located on the default host.

//

Hashtable env = new Hashtable();

env.put(Context.SECURITY_PRINCIPAL, "anonymous");

env.put(Context.SECURITY_CREDENTIALS, "anonymous");

env.put(Context.PROVIDER_URL, url);
env.put(Context.INITIAL_CONTEXT_FACTORY,
"fiorano.jms.runtime.naming.FioranoInitialContextFactory");



env.put(FioranoJNDIContext.ENABLE_AUTO_REVALIDATION, "true");

InitialContext ic = new InitialContext(env);

System.out.println("Created InitialContext :: " + ic)
```

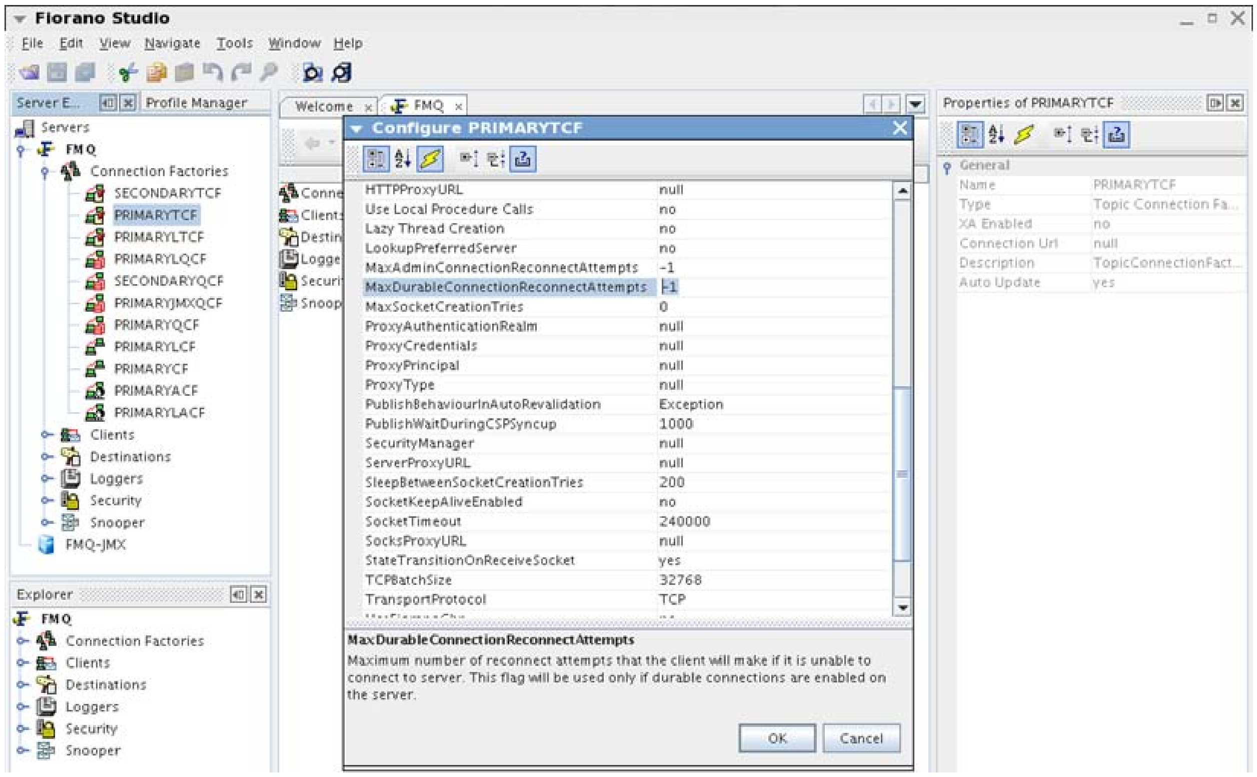
5.3 Setting MaxDurableConnectionReconnectAttempts in Server

This parameter denotes the maximum number of reconnection attempts made by a client if it is unable to connect to server. This flag will be used only if durable connections are enabled on the server. If set to a finite positive number, the client will try to revalidate the durable connection the specified number of times after which it will stop revalidating the connection and throw an exception. By default, this number is set to '-1'. This -1 denotes that the client will try indefinitely to reconnect to the server.

5.3.1 Online Mode

Steps to set MaxDurableConnectionReconnectAttempts in the metadata of a ConnectionFactory, say primaryQCF, are listed below.

1. Launch the **admin studio** and make sure that FioranoMQ Server is running.
2. **Login** to the server using **FioranoMQ login** and go to **ConnectionFactories**.
3. Right-click on **ConnectionFactories** and click on the **Edit** option.
4. Enter a valid value for **MaxDurableConnectionReconnectAttempts** and click **OK**.
5. **Save** the Configuration.



HTTPProxyURL	null
Use Local Procedure Calls	no
Lazy Thread Creation	no
LookupPreferredServer	no
MaxAdminConnectionReconnectAttempts	-1
MaxDurableConnectionReconnectAttempts	-1
MaxSocketCreationTries	0
ProxyAuthenticationRealm	null
ProxyCredentials	null
ProxyPrincipal	null
ProxyType	null
PublishBehaviourInAutoRevalidation	Exception
PublishWaitDuringCSPSyncup	1000
SecurityManager	null
ServerProxyURL	null
SleepBetweenSocketCreationTries	200
SocketKeepAliveEnabled	no
SocketTimeout	240000
SocksProxyURL	null
StateTransitionOnReceiveSocket	yes
TCPBatchSize	32768
TransportProtocol	TCP

MaxDurableConnectionReconnectAttempts
Maximum number of reconnect attempts that the client will make if it is unable to connect to server. This flag will be used only if durable connections are enabled on the server.

General	
Name	PRIMARYTCF
Type	Topic Connection Fa...
XA Enabled	no
Connection Url	null
Description	TopicConnectionFact...
Auto Update	yes

5.4 Setting MaxDurableConnectionReconnectAttempts from Client Application

MaxDurableConnectionReconnectAttempts can also be specified from the client application:

In the client application, prior to creating the **InitialContext** and lookup the **ConnectionFactory**, add the line below to the code to set the environment property for **MaxDurableConnectionReconnectAttempts**.

```
..

env.put(Context.PROVIDER_URL, Jndi.PROVIDER_URL);

env.put(Context.INITIAL_CONTEXT_FACTORY, Jndi.INITIAL_CONTEXT_FACTORY);

env.put("BackupConnectURLs", Jndi.BACKUP_CONNECT_URLS);



env.put("MaxDurableConnectionReconnectAttempts", "3"); //Newly added Line
```

Chapter 6: Configuring Message Store

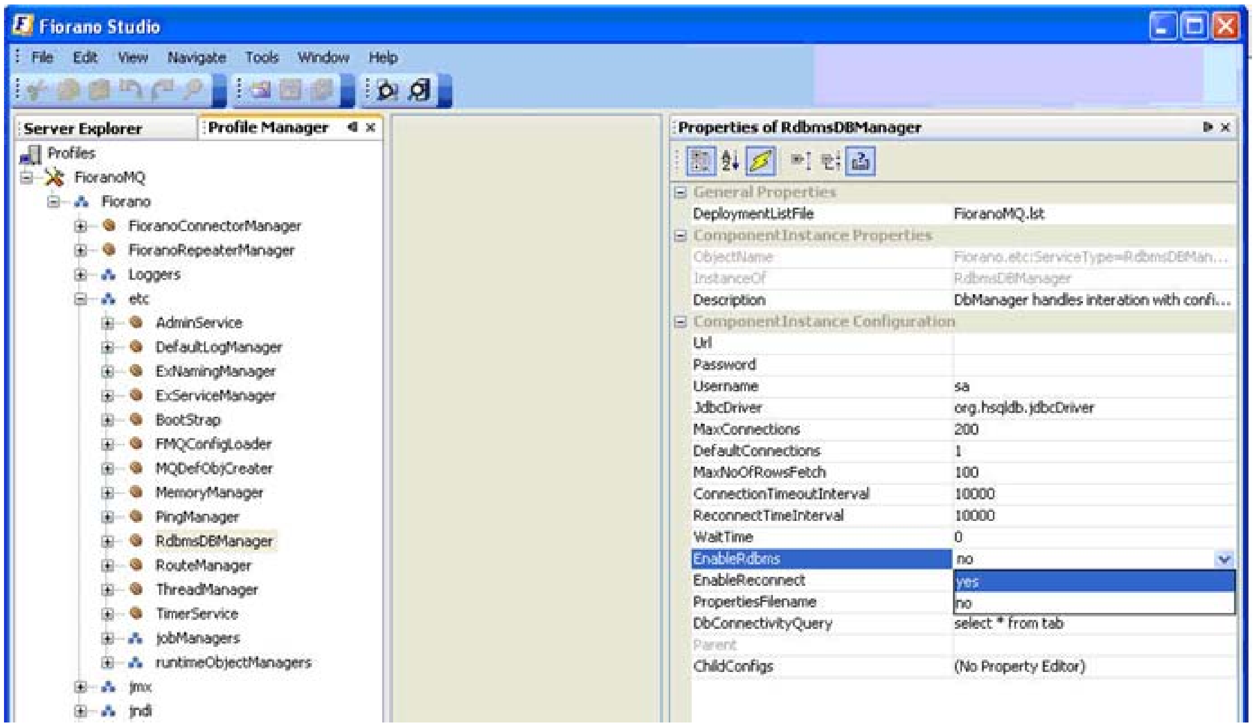
Note:

- Add the corresponding JAR files for the database to be configured for the FioranoMQ Server, to the Configuration file of the server, fmq.conf. This file is present at fmq_installation_dir\fmq\bin.
- To start the FioranoMQ Server with one of the database, run create-database.bat/sh file after adding the required JAR to the classpath of *create*-database.conf.

6.1 Enabling RDBMS

By default, RDBMS Support is turned Off. RDBMS can be enabled by following the steps below:

1. Open the profile for offline editing through the **Profile Manager** using **Studio** as explained in section 4.1.
2. Navigate to bean **Fiorano -> etc -> RdbmsManager** and set the property **EnableRdbms** to **yes** as shown in the figure below:



Properties of RdbmsDBManager

General Properties	
DeploymentListFile	FioranoMQ.lst
ComponentInstance Properties	
ObjectName	Fiorano.etc:ServiceType=RdbmsDBMan...
InstanceOf	RdbmsDBManager
Description	DbManager handles interation with confi...
ComponentInstance Configuration	
Url	
Password	
Username	sa
JdbcDriver	org.hsqldb.jdbcDriver
MaxConnections	200
DefaultConnections	1
MaxNoOfRowsFetch	100
ConnectionTimeoutInterval	10000
ReconnectTimeInterval	10000
WaitTime	0
EnableRdbms	no
EnableReconnect	yes
PropertiesFilename	no
DbConnectivityQuery	select * from tab
Parent	
ChildConfigs	(No Property Editor)

3. Edit the configuration specifying the JDBC parameters for the database. (A sample configuration for some common databases can be found in the next section.)
4. Right-click on the **FioranoMQ** node and select **Save** from the pop-up menu.

6.2 Sample Configuration

6.2.1 DB2

URL: jdbc:db2://<DBServer>:6789/sample

JdbcDriver: COM.ibm.db2.jdbc.net.DB2Driver

Username: <username>

Password: <password>

PropertiesFilename: jdbc_db2.cfg

MaxConnections: 200

6.2.2 Oracle

URL: jdbc:oracle:thin:@<machine Ip>:1521:<sid>

JdbcDriver: oracle.jdbc.driver.OracleDriver

Username: <username>

Password : <password>

PropertiesFilename: jdbc_oracle.cfg

MaxConnections: 200

6.2.3 MSSQL

URL:
jdbc:microsoft:sqlserver://<machineName>:1433;SelectMethod=Cursor;databaseName=<database_name>

If using the Microsoft SQL 2005 or later Server driver, the URL is:

URL:
jdbc:sqlserver://<machineName>:1433;SelectMethod=Cursor;databaseName=<database_name>

JdbcDriver: com.microsoft.jdbc.sqlserver.SQLServerDriver

Ifusing the Microsoft SQL 2005 or later Server driver, the JdbcDriver is:

JdbcDriver: com.microsoft.sqlserver.jdbc.SQLServerDriver

Username: <username>

Password: <password>

PropertiesFilename: jdbc_mssqls.cfg

MaxConnections: 200

Note : If using the MS SQL 2005 or later Server drive, the DB_TABLE_NOT_FOUND should be changed from **jdbc_mssqls.cfg** to **S0002**, or else the default value of **42S02** will remain.

6.2.4 MySQL

URL: jdbc:mysql://localhost/mysql

JdbcDriver: com.mysql.jdbc.Driver

Username: <username>

Password: <password>

PropertiesFilename: jdbc_mysqls.cfg

MaxConnections: 200

6.2.5 Cloudscape

URL: jdbc:cloudscape:mydb;create=true

JdbcDriver : COM.cloudscape.core.JDBCDriver

Username: <username>

Password: <password>

PropertiesFilename: jdbc_cloudscape.cfg

MaxConnections: 200

6.3 Additional Configuration

The Database Driver used should be made available in the **classpath** when launching the server. This can be done by editing the respective configuration files in launch scripts - that is, **fmq\bin\fmq.conf, fmq\bin\ClearDB.conf**, and **fmq\bin\create-database.conf**, respectively.

6.4 Creating a Default Database

FioranoMQ comes with a script (**fmq/bin/ create-database**) that allows the creation of a default database for the server. This script accepts input from the database in the following ways:

- Through Command Line Parameters
- Through a pre-configured Fiorano Profile

6.4.1 Command Line Parameters

The following command lines are accepted through the create-database script

driver

This parameter specifies the **class name** of the driver class for the database.

url

This parameter specifies the **URL** of the database.

username

This parameter specifies the **login name** of the User that the **database script** uses to connect to the database.

password

This parameter specifies the **Password** of the User that the **database script** uses to connect to the database.

dataTypesFileName

This parameter specifies the complete **path name** of the file that contains the mapping of the Java data 'types' to SQL data 'types'. Please see the Sample configuration section for various databases to know the complete property file name for a specific database. If a different database is being used, please send a mail to: presales@fiorano.com.

databaseType

This parameter specifies the name of the database being used. Valid database 'types' include oracle, mssql, mysql, db2, and cloudscape. If a different database is being used, please send a mail to: presales@fiorano.com.

6.4.2 Pre-configured Profile

Configure the server to enable RDBMS as explained in the section on Enabling RDBMS.

Run the create-database script and specify the profile directory. Enter the 'type' of database being used.

create-database -profile <FMQProfile> -dataBaseType <DataBaseType>

Note:

- Ensure that the database driver classes are present in the classpath.
- FioranoMQ ships the library that contains the jdbc driver for HSQL.

Profile Directory is relative to the profiles directory.

If profile is not specified, the default profile, which is **FioranoMQ**, is used.

6.5 Clearing a Database

To clear the database run the ***ClearDB.bat*** file from a Windows platform or the ***ClearDB.sh*** file from a UNIX platform. This script accepts the name of the profile for the message store implementation as a system variable. Both, **RDBMS** and **File-based** databases are cleared.

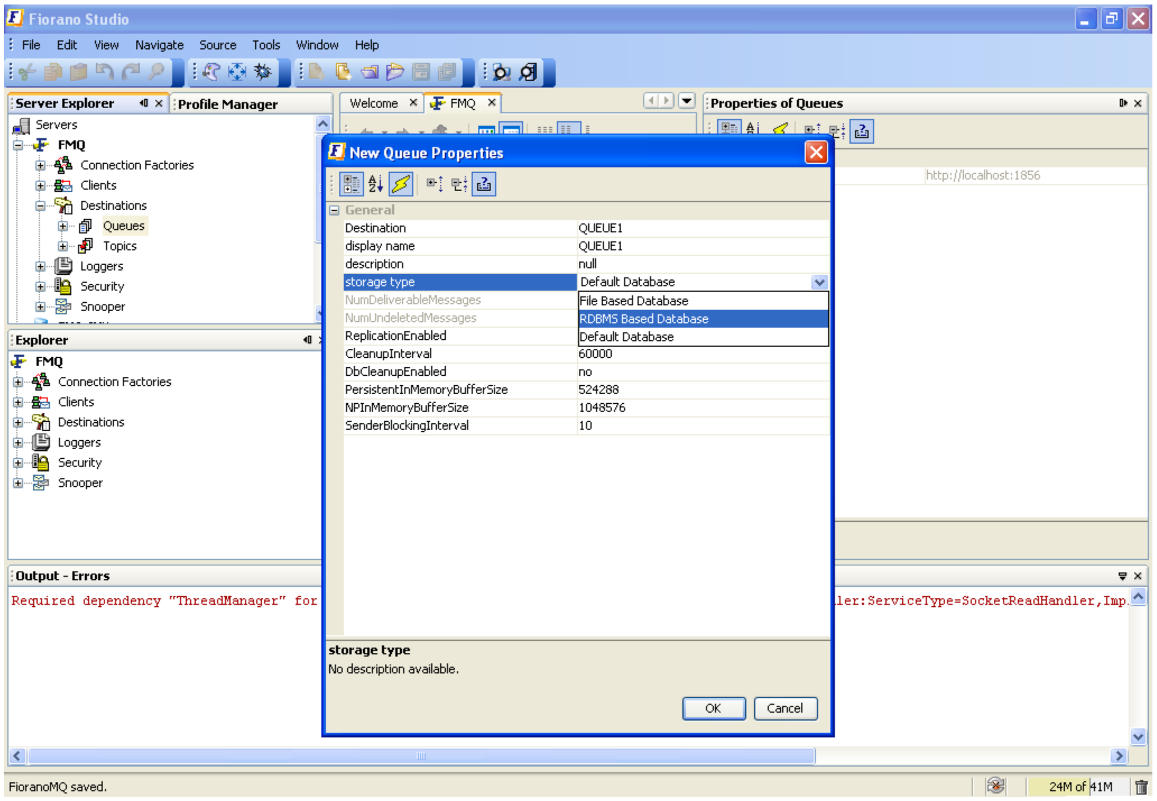
6.6 Creating a Destination on RDBMS

FioranoMQ provides the option to configure the store of a destination and set it to either a **File-based** or a **RDBMS based** database.

The administrator is free to use both databases by creating destinations on Files and on RDBMS. To create RDBMS based destinations, follow the steps below:

1. Launch Studio and right-lick and select the server. Select **Login** from the pop-up menu.
2. To create a new destination, navigate to the **Destinations** node.

3. Right-click on either '**Queue'** or on '**Topic'** and select **Add Queue** or **Add Topic**. The dialog box that appears displays properties for the destination. Select storage type as RDBMS Based Destination, as shown in the figure below:



Destination	QUEUE1
display name	QUEUE1
description	null
storage type	Default Database
NumDeliverableMessages	File Based Database
NumUndeletedMessages	RDBMS Based Database
ReplicationEnabled	Default Database
CleanupInterval	60000
DbCleanupEnabled	no
PersistentInMemoryBufferSize	524288
NPInMemoryBufferSize	1048576
SenderBlockingInterval	10

storage type
No description available.

```
Required dependency "ThreadManager" for
ler:ServiceType=SocketReadHandler,Imp.
```

FioranoMQ saved.

4. Click on the **OK** button.

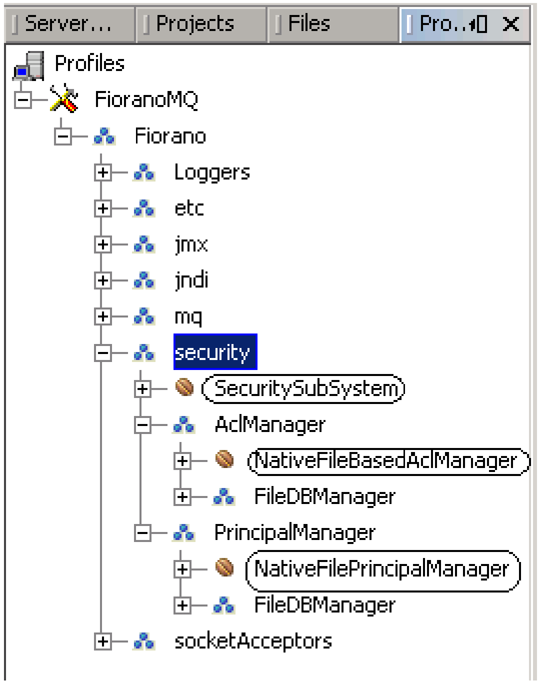
Chapter 7: FioranoMQ Security

For information about configuring profiles through text-based files, refer to FioranoMQ Getting Started.

7.1 Security Related MBeans

Security related components are found in the default MQ profiles located under the **Fiorano->Security** domain. The **Object Names** for these components are:

- Fiorano.security:ServiceType=RealmManager,Name=SecuritySubSystem
- Fiorano.security.AclManager:ServiceType=AclManager,Impl=FILE,Name=NativeFileBasedAclManager
- Fiorano.security.PrincipalManager:ServiceType=PrincipalManager,Impl=FILE,Name=NativeFilePrincipalManager

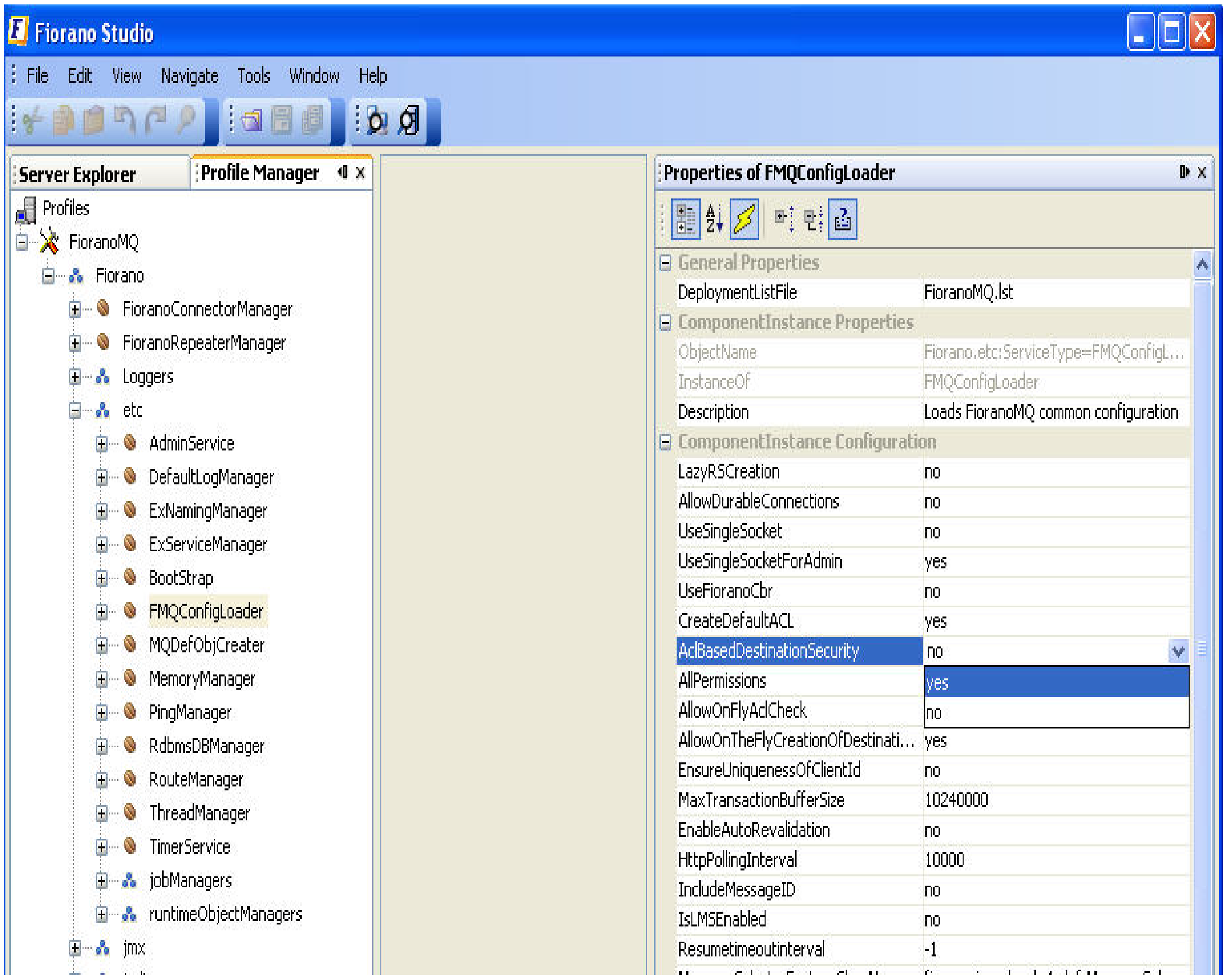


The figure above shows the position of these components in the component tree as seen by an off-line configuration tool (the **Profile Manager**).

7.2 How to Enable ACL Based Security

By default, ACL based security is turned OFF in FioranoMQ, but can be turned ON, as shown in the steps below:

1. Open the profile for off-line editing through the **Profile Manager** as explained in section 4.8.1 How to Enable Pinging
2. Go to **Fiorano -> etc -> FMQConfigLoader**.
3. In the property panel change the value of the **AclBasedDestinationSecurity** property to '**yes**' as shown in the figure below:



- Profiles
  - FioranoMQ
    - Fiorano
      - FioranoConnectorManager
      - FioranoRepeaterManager
      - Loggers
      - etc
        - AdminService
        - DefaultLogManager
        - ExNamingManager
        - ExServiceManager
        - BootStrap
        - FMQConfigLoader
        - MQDefObjCreater
        - MemoryManager
        - PingManager
        - RdbmsDBManager
        - RouteManager
        - ThreadManager
        - TimerService
        - jobManagers
        - runtimeObjectManagers
      - jmx

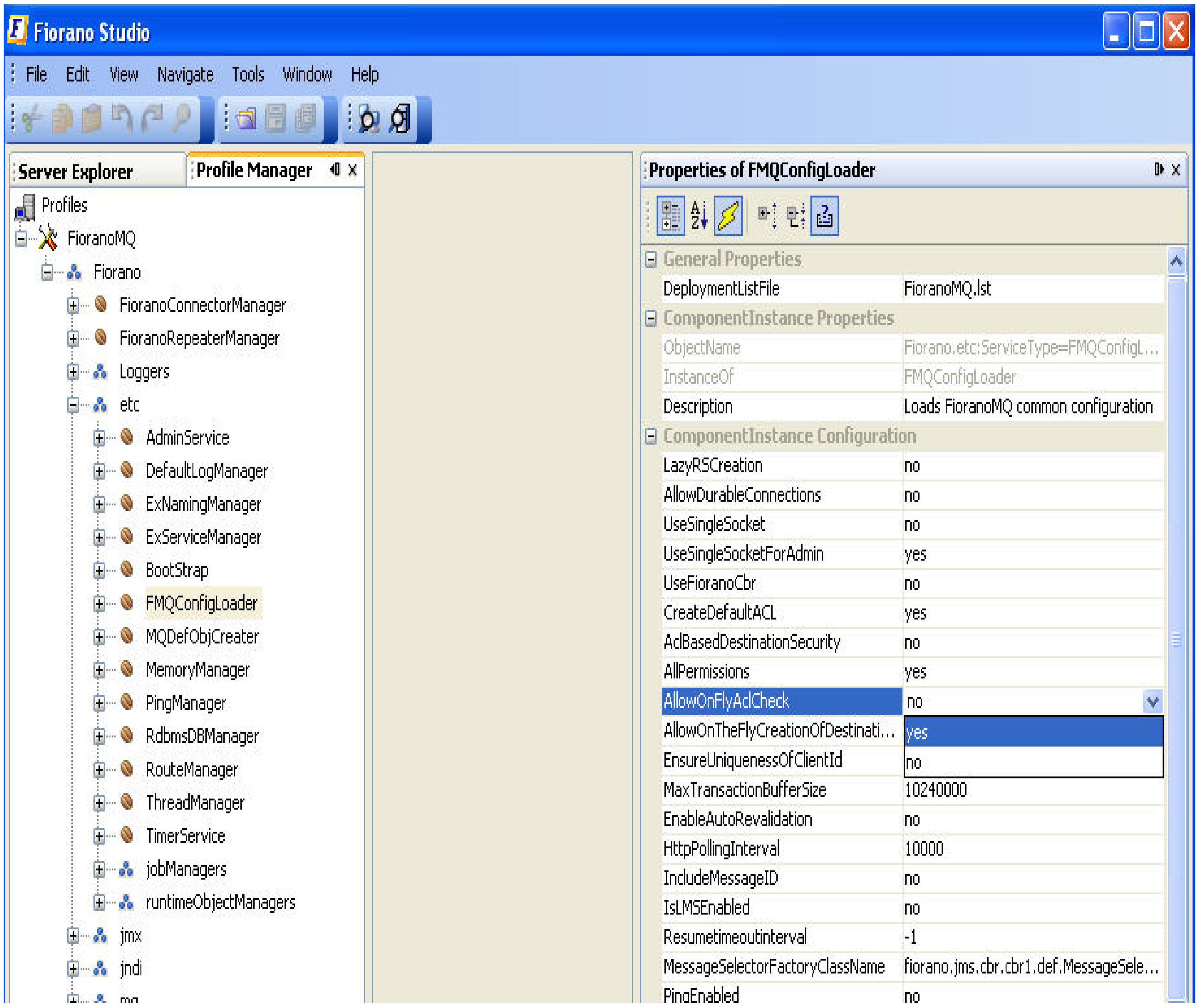
General Properties	
DeploymentListFile	FioranoMQ.lst
ComponentInstance Properties	
ObjectName	Fiorano.etc:ServiceType=FMQConfigL...
InstanceOf	FMQConfigLoader
Description	Loads FioranoMQ common configuration
ComponentInstance Configuration	
LazyRSCreation	no
AllowDurableConnections	no
UseSingleSocket	no
UseSingleSocketForAdmin	yes
UseFioranoCbr	no
CreateDefaultACL	yes
AclBasedDestinationSecurity	no
AllPermissions	yes
AllowOnFlyAclCheck	no
AllowOnTheFlyCreationOfDestinati...	yes
EnsureUniquenessOfClientId	no
MaxTransactionBufferSize	10240000
EnableAutoRevalidation	no
HttpPollingInterval	10000
IncludeMessageID	no
IsLMSEnabled	no
Resumetimeoutinterval	-1

4. Right-click on the FioranoMQ node and select **Save** from the pop-up menu.

7.3 How to Turn ON ACL Checks

By default, ACL's are checked only time of performing an action such as creating a publisher/subscriber on a topic. If an ACL is modified, clients connected to it are not affected. To get connected clients to check ACL when modified, the steps below may be followed:

1. Open the profile for off-line editing through the **Profile Manager** using Studio, as explained in 4.8.1 How to Enable Pinging
2. Go to **Fiorano -> etc -> FMQConfigLoader**.
3. In the property panel change the value of the **AllowOnTheFlyAclCheck** property to '**yes'** as shown in the figure below:



Profiles
- FioranoMQ
  - Fiorano
    - FioranoConnectorManager
    - FioranoRepeaterManager
    - Loggers
    - etc
      - AdminService
      - DefaultLogManager
      - ExNamingManager
      - ExServiceManager
      - BootStrap
      - FMQConfigLoader
      - MQDefObjCreater
      - MemoryManager
      - PingManager
      - RdbmsDBManager
      - RouteManager
      - ThreadManager
      - TimerService
      - jobManagers
      - runtimeObjectManagers
    - jmx
    - jndi

General Properties	
DeploymentListFile	FioranoMQ.lst
ComponentInstance Properties	
ObjectName	Fiorano.etc:ServiceType=FMQConfigL...
InstanceOf	FMQConfigLoader
Description	Loads FioranoMQ common configuration
ComponentInstance Configuration	
LazyRSCreation	no
AllowDurableConnections	no
UseSingleSocket	no
UseSingleSocketForAdmin	yes
UseFioranoCbr	no
CreateDefaultACL	yes
AclBasedDestinationSecurity	no
AllPermissions	yes
AllowOnFlyAclCheck	no
AllowOnTheFlyCreationOfDestinati...	yes
EnsureUniquenessOfClientId	no
MaxTransactionBufferSize	10240000
EnableAutoRevalidation	no
HttpPollingInterval	10000
IncludeMessageID	no
IsLMSEnabled	no
Resumetimeoutinterval	-1
MessageSelectorFactoryClassName	fiorano.jms.cbr.cbr1.def.MessageSele...
PingEnabled	no

4. Right-click on the FioranoMQ node and select **Save** from the pop-up menu.

Note:

The **AllowOnFlyAclCheck** flag works for all permissions except when:

1. A publisher is publishing non-persistent messages on a topic.

2. The permission to create publisher should be revoked for a topic.
3. No exception is thrown even though the User is not allowed to publish since messages are sent in batch mode.

Work Around 1:

For NP messages, batching is enabled by default which leads to the behavior explained above. To view this Exception at the 'send' API location, set the BatchingEnabled parameter in the ConnectionFactory to 'FALSE'.

Work Around 2:

Add the following line to the client code environment while performing the lookup function:

```
env.put("BatchingEnabled", "false")
```

Here '**env'** is the environment passed while performing a JNDI lookup. This will disable batching for that particular client.

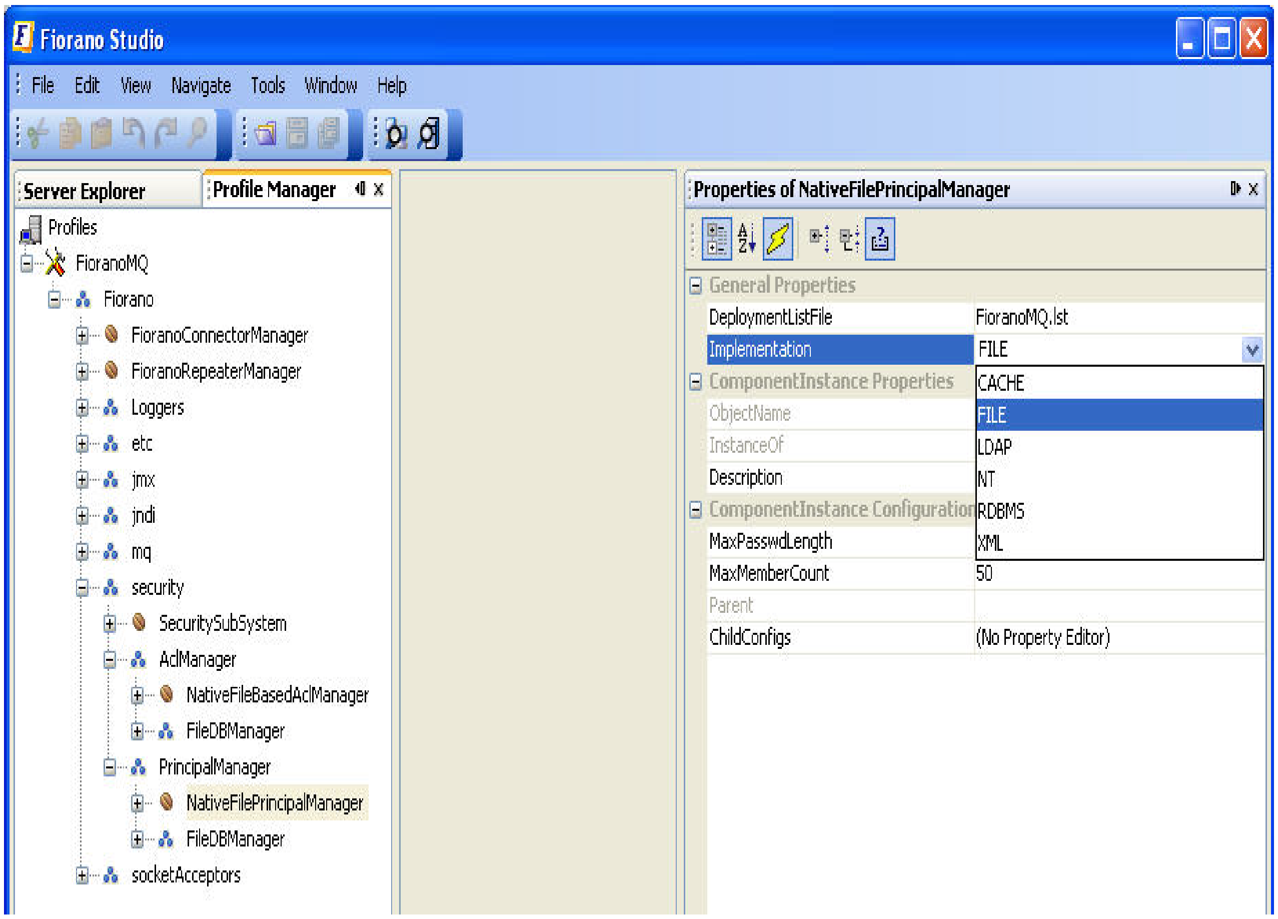
7.4 Modifying ACLManager Implementation

1. Open the profile for off-line editing through the **Profile Manager** as explained in section 4.8.1
2. Browse to reach the node **Fiorano -> security -> AclManager**. Click on the **current ACL Manager MBean**.
3. In the properties panel, click on the value of the **Implementation** property and choose a value from the drop-down menu.
4. Right-click on the FioranoMQ node and select **Save** from the pop-up menu.

7.5 Modifying Principal Manager Implementation

1. Open the profile for off-line editing through the **Profile Manager** using Studio as explained in section 4.8.1
2. Browse to reach **Fiorano -> security -> PrincipalManager** and click on the current Principal Manager Mbean as shown in the figure below.

3. In the property panel, click on the value of the **Implementation** property and choose a value from the drop-down list.



Properties of NativeFilePrincipalManager

General Properties	
DeploymentListFile	FioranoMQ.lst
Implementation	FILE
ComponentInstance Properties	
ObjectName	
InstanceOf	
Description	
ComponentInstance Configuration	
MaxPasswdLength	
MaxMemberCount	50
Parent	
ChildConfigs	(No Property Editor)

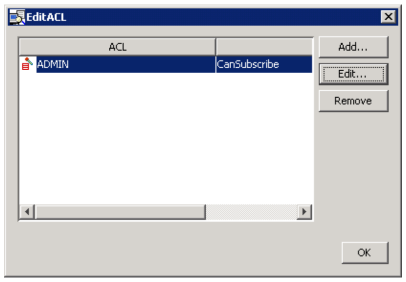
4. Right-click on the FioranoMQ node and select **Save** from the pop-up menu

7.6 Editing Destination Level Security Through ACL's

The administrator can grant users access permissions to work on different destinations (Topics and queues). Permissions for Users and User Groups may be edited by performing the steps below:

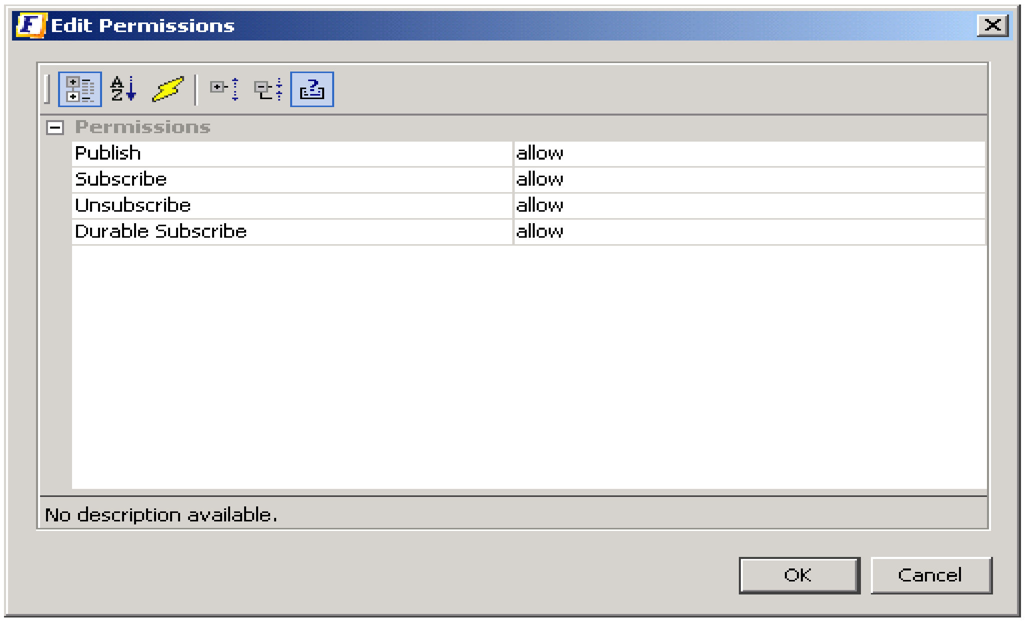
1. Launch the **Studio** and click on the **Server Explorer** pane. Right-click on the desired server and select **Login** from the pop-up menu.
2. Navigate to the required destination through **Destinations > Topics/Queues**.

3. Right-click the required **topic/queue** and select the **EditACL** option from the pop-up menu. The **EditACL** dialog box is displayed, as shown in the figure below:



ACL	
ADMIN	CanSubscribe

4. Permissions for a new principal may be added by clicking on the **Add** button.
5. An existing entry for a principal can be removed by clicking on the **Remove** button.
6. Select the ACL Entry (for any principal) in the dialog box and click the **Edit** button. The **Edit Permissions** dialog box, shown below, will be displayed.



Permissions

Publish	allow
Subscribe	allow
Unsubscribe	allow
Durable Subscribe	allow

No description available.

7. Modify the permissions for various actions such as Publish, Subscribe, Unsubscribe, and Durable Subscribe and click on the **OK** button.

7.7 Configuring NT Based security

This section describes how to set up and configure the FioranoMQ Windows NT security realm (Fiorano NT Realm) for the FioranoMQ server. Fiorano NT Realm works both on Windows NT 4.0 and Windows 2000.

7.7.1 Pre-requisites

Fiorano NT Realm requires that the FioranoMQ server is run by a Windows administrative User able to read security-related data from the Windows NT Domain Controller. To use the Fiorano NT Realm, FioranoMQ is run on the Windows NT domain.

To manage User and User Group information, the FioranoMQ server must be able to make system calls to the Windows NT computer on which the FioranoMQ server runs. To perform authentication, FioranoMQ needs the privileges that would allow it to communicate with the Primary Domain Controller.

7.7.2 Setting up

1. Launch the **Fiorano Studio**. Configure the NT based **PrincipalManager** as explained in the section Modifying Principal Manager Implementation.
2. Right-click on the **FioranoMQ domain** from the **Server Explorer** and select the **Save** option from the pop-up menu.

Configuring Windows NT

1. Login to Windows NT using Administrator permissions. Navigate to **User Manager** in the **Administrative Tools** program group from the Windows NT machine on which FioranoMQ is installed.
2. Select a User that is enabled to run the **FioranoMQ server**. Choose **User Rights** from the **Policies** menu.
3. Select the **Show Advanced User Rights** from the **Rights** list and click **Add**. Enter the **name of the User** who is to execute **FioranoMQ**.
4. Select **Replace Process Level Token** from the **Rights** list. Click **Add** and enter the name of the ser who is to execute **FioranoMQ**.
5. Restart the System. The new permissions for the User take effect.

Configuring Windows 2000

1. **Login** with **Administrator permissions** onto the Windows 2000 machine where the FioranoMQ Server is installed.
2. Open **Control Panel** > **Administrative Tools >Local Security Policy**.
3. Open the **Local Policies** tree.
4. Click **User Rights** Assignments.
5. On the right-hand pane, right-click **Act.**
6. Select **Security** from the menu.

7. On the next panel, click **Add** and choose the **name of the User** who is to execute **FioranoMQ**.
8. Click on the **OK** button. Restart the System. The new permissions for the User take effect.

Additional Configuration – Adding FioranoMQ Users to Administrators Group

In the NT Principal Manager, only users registered with the Administrators group have the rights to open/create AdminConnection. Other Users may be given these rights by adding/registering them with the default Administrators' group as explained below:

1. Open **Control Panel > Administrative Tools >Users and Passwords**.
2. Browse to reach **Local Users and Groups > Groups > Administrators**.
3. Click on **Add** to display a list of all the Users that exist in the WinNT Realm. Users can be included in the Administrators group by adding them from the list.

The User admin used by default to create admin connections is not a member of the Administrators group for the FioranoMQ NT Realm. In order to use FioranoMQ default admin tools and APIs, the admin User must register with the Administrators' group.

Verifying

When starting the FioranoMQ Server after installing and configuring FioranoNTRealm, the verification checks listed below need to be performed:

1. Open a **Command-shell**. Navigate to the *Samples\Realm* folder.
2. Compile the test case by using ***compile-client TestFioranoNTRealm.java***.
3. Run the test case by using ***run-client TestFioranoNTRealm***. On successful execution, the test case displays a message.
4. Run the **AdminGUI**. Check that the list of Windows NT Users and Groups are displayed on the User and Group Panel.

Limitations

Below is a list of those 'aspects'/'functions' not supported by the FioranoMQ 9 version of the Fiorano NT Realm. However, these functions are supported within the File-based Implementation of the FioranoMQ Realm:

1. A **Group** cannot have a Group as a member.
2. Changing the password for a User through FioranoNTRealm API is not allowed.

Troubleshooting FioranoMQ NT Realm

The most common configuration problems encountered with Fiorano NT Realm are related with Windows NT policies and specifically with the User who runs the FioranoMQ server. The User requires special permissions to access the Windows NT domain. The steps for granting these permissions are in configuration instructions as mentioned in the section 7.7.2 Setting up. Another common problem is the inability of the FioranoMQ server to load the fioranorealm.dll file. If FioranoMQ is unable to load the fioranorealm.dll, the following message is displayed:

```
java.lang.UnsatisfiedLinkError: no fioranorealm in java.library.path
```

```
at java.lang.ClassLoader.loadLibrary(ClassLoader.java:1312)

at java.lang.Runtime.loadLibrary0(Runtime.java:749)

at java.lang.System.loadLibrary(System.java:820)

at fiorano.jms.realm.principal.nt.FioranoNTManager.init(FioranoNTManager.java:82)

at fiorano.jms.realm.principal.nt.FioranoNTManager.(FioranoNTManager.java:51)

at fiorano.jms.realm.principal.nt.PrincipalManagerImpl.startup(PrincipalManager-

Impl.java:61)

at fiorano.jms.realm.RealmManagerImpl.startup(RealmManagerImpl.java:77)

at fiorano.jms.ex.Executive.startup(Executive.java:647)

at fiorano.jms.ex.Kernel.startup(Kernel.java:61)

at fiorano.jms.ex.fmpmain.main(fmpmain.java:60)

fiorano.jms.common.FioranoException: REALM_NOT_SUPPORTED :: NT realm support is not
available
```

7.8 RDBMS Realm

7.8.1 Setting up

1. Open the profile for off-line editing through the **Profile Manager** as explained in section 4.8.1 How to Enable Pinging.
2. Modify Principal Manager [Modifying Principal Manager Implementation] and ACL Manager implementation [Modifying ACLManager Implementation] to RDBMS
3. Configure these components as per the database used. A sample configuration for some common databases is provided in later sections.
4. Right-click on the FioranoMQ node and select **Save** from the pop-up menu.

7.8.1.1 Additional Configuration

The database driver needs to be added to the Container classpath.

To force the FioranoMQ server to create default Destinations and Users in the recently configured RDBMS server, the existing database is to be cleared (run folder of profile) prior to restarting the server.

7.8.1.2 Sample Configurations

The list below provides sample configurations for various databases. These parameters can be specified for the ACLManager as well as for the PrincipalManager.

7.8.1.2.1 Oracle

URL: jdbc:oracle:thin:@164.164.128.113:1521:orcl

Database Driver: oracle.jdbc.driver.OracleDriver

Username: scott

Password: tiger

7.8.1.2.2 MySQL

URL: jdbc:mysql://localhost/mysql

DatabaseDriver: com.mysql.jdbc.Driver

Username: <user name>

Password: <password>

7.8.1.2.3 HSQL

URL: jdbc:hsqldb:d:\FMQDB

DatabaseDriver: org.hsqldb.jdbcDriver

Username: sa

Password: <password>

7.8.1.2.4 MSSQL

URL: jdbc:microsoft:sqlserver://qalab01:1433

DatabaseDriver: com.microsoft.jdbc.sqlserver.SQLServerDriver

Username: sa

Password: <password>

Note: The MS-SQL driver has to be added to the Container classpath(msutil.jar,mssqlserver.jar,msbase.jar)

7.8.1.2.5 DB2

URL: jdbc:db2://localhost:7777/sample

DatabaseDriver: COM.ibm.db7.jdbc.net.DB2Driver

Username: user

Password: passwd

The parameter named PropertiesFile should point to principalsqlstatements properties when configuring PrincipalManager, and to aclsqlstatements properties when configuring ACLManager. These files can be found in the conf folder of the profile being used.

7.8.1.3 Verifying

Use a Query tool provided by the database vendor (SQLWorksheet for Oracle) and verify the creation of the following tables with their default values:

7.8.1.3.1 Principal Manager

TableName - users (stores all Users related information)

TableName - groupmembers (stores all Group related information)

7.8.1.3.2 ACL Manager

TableName - aclentries (stores ACL Entries for all users)

The following is a sample SQL query executed in a SQLWorksheet:

```
SQLWKS> select * from users;

SQLWKS> select * from aclentries;
```

7.9 LDAP Security Realm

1. Open the profile for off-line editing through the **Profile Manager** using Studio by clicking on the **Profile Manager** pane, explained in section 4.8.1 How to Enable Pinging
2. Modify the Implementation property of the ACL Manager and the Principal Manager to LDAP. For more information on how to modify ACL Manager and Principal Manager refer to the section Modifying ACL Manager Implementation and Modifying Principal Manager Implementation.
3. Configure the Principal Manager as per the Directory Server in use. A sample configuration for the Netscape Directory Server is shown in the figure Directory Server Settings.
4. Right-click on the FioranoMQ node and select **Save** from the pop-up menu.

7.9.1 Sample Configuration – Netscape Directory Server

Setting the Name

The 'name' is the name of the admin of the LDAP server, since the Initial Context may only be started by the Admin.

PRINCIPAL = uid=admin, ou=Administrators, ou=TopologyManagement, o=NetscapeRoot

Setting the password

Enter the password for the Admin of the LDAP Server with whom a connection is to be make as shown in the Figure below:

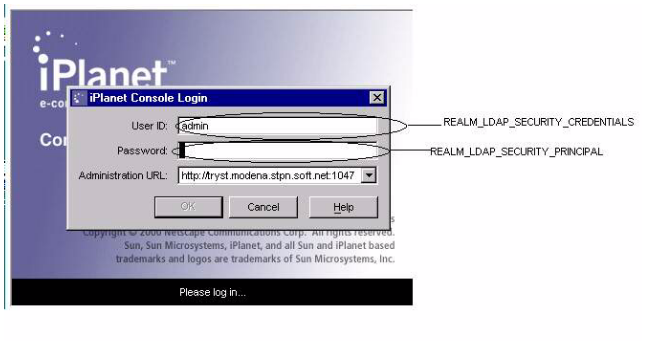

iPlanet
iPlanet Console Login
User ID:
admin
Password:
Administration URL:
http://tryst.modena.stpn.soft.net:1047
OK
Cancel
Help
REALM_LDAP_SECURITY_CREDENTIALS
REALM_LDAP_SECURITY_PRINCIPAL
Sun, Sun Microsystems, iPlanet, and all Sun and iPlanet based trademarks and logos are trademarks of Sun Microsystems, Inc.
Please log in...

Figure: iPlanet Console Login Dialog Box

LDAP Initial Context Factory

The Initial Context Factory to be used, corresponding to the directory server.

LdapInitialCtxFactory = com.sun.jndi.ldap.LdapCtxFactory

LDAP Provider URL

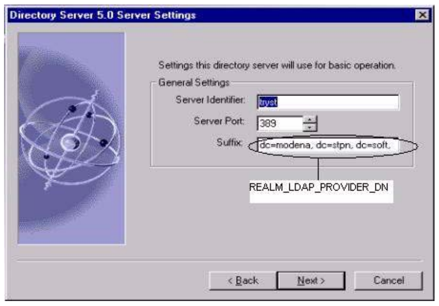
Is set in accordance with the Directory Server being used.

LdapProviderUrl = ldap://ldapserver:389

LDAP Provider DN

This is to be set to the suffix variable set up while installing the LDAP Server as shown in the figure Directory Server Settings:

: LdapProviderDn = dc=modena, dc=stpn, dc=soft, dc=net



Settings this directory server will use for basic operation.

General Settings

Server Identifier: tryst

Server Port: 389

Suffix: dc=modena, dc=stpn, dc=soft,

REALM_LDAP_PROVIDER_DN

Figure: Directory Server Settings

LDAP security authentication

Set this variable to:

LdapSecurityAuthentication = Simple

LDAP User and Group Object classes

7.9.2 Sample Configuration – ApacheDS1.5.4

Note: The steps mentioned here require the installation of the Apache Directory Studio.

7.9.2.1 Setting up the Directory Service

To setup the directory service, the steps below are to be performed:

1. Stop any running instance of ApacheDS.
2. Take a backup of server.xml

   /var/lib/apacheds-1.5.4/default/conf/server.xml (DEFAULT PATH. If the DS instances were installed in a location different, server.xml will be available inside the directory at that location.)
3. Modify server.xml by adding the line below within the tag *</partitions> ... </partitions>*

   <jdbmPartition id="fiorano" cacheSize="100" suffix="o=fiorano,c=US" optimizerEnabled="true" syncOnWrite="true"/>
4. Run apacheds

   /etc/init.d/apacheds start
5. Login through the Apache Directory Studio.

User	: uid=admin,ou=system.	(Default)
Password	: secret.	(Default)
6. Import the LDIF content below using Apache Directory Studio. (Menu: *LDAP -> New LDIF File*)

   ```
   dn: o=fiorano,c=us
   ```

```
objectclass: top

objectClass: organization

o: fiorano

dn: cn=FMQServerConfigFiles,o=fiorano,c=us

objectclass: top

objectClass: organizationalRole

cn: FMQServerConfigFiles

dn: cn=FioranoMQUsers,o=fiorano,c=us

objectClass: top

objectClass: organizationalRole

cn: FioranoMQUsers


dn: cn=FioranoMQGroups,o=fiorano,c=us

objectClass: top

objectClass: organizationalRole

cn: FioranoMQGroups


dn: cn=ACL,o=fiorano,c=US

objectclass: top

objectClass: organizationalRole

cn: ACL


dn: cn=FMQRoot,o=fiorano,c=us

objectClass: inetOrgPerson

objectClass: organizationalPerson

objectClass: person

objectClass: top

cn: FMQRoot
```

```
cn: system administrator

sn: administrator

displayname: Directory Superuser

userpassword:: c2VjcmV0
```

7. Re-login through Apache Directory Studio to see the added children.

7.9.2.2 Setting up the profile for use with ApacheDS1.5.4

Note: Make sure that the steps mentioned in section 7.9 have been completed before moving on to the steps listed below:

1. Open the profile for off-line editing through the Profile Manager using Studio, as explained in section 4.8.1 How to Enable Pinging
2. Reset all properties except the LdapProviderUrl to their original values.
3. In the LDAP Provider URL, the port number is **10389** and the ip address is that of the server that running ApacheDS.

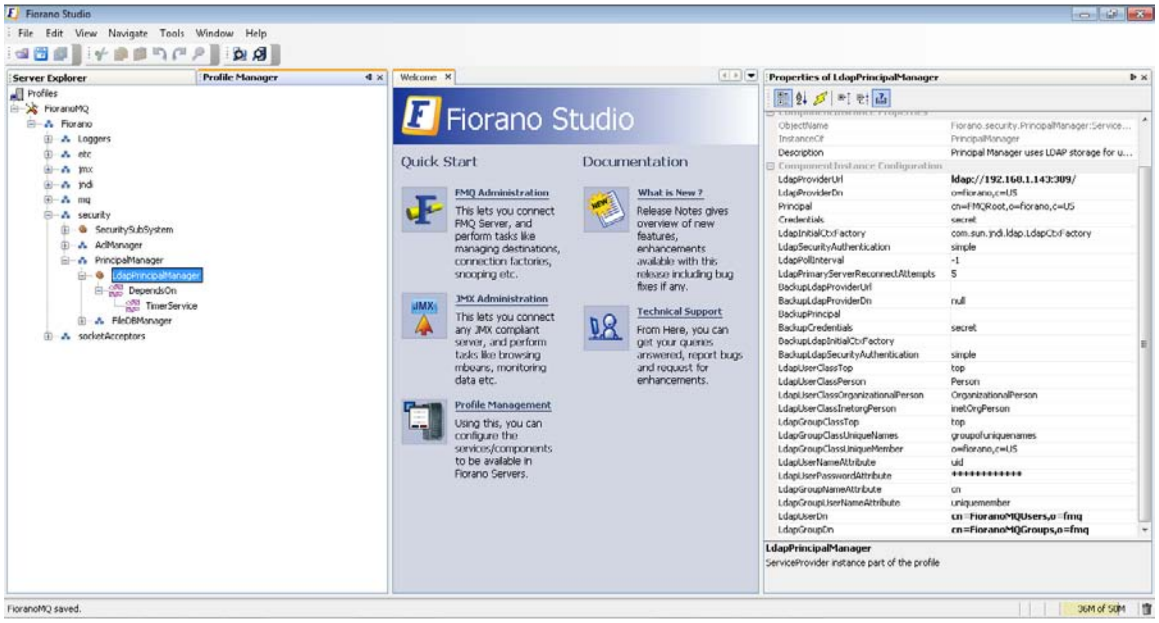
7.9.3 Sample LDAP Configuration for ACLs, Users and Groups

7.9.3.1 Configuration for Users and Groups

Here is an example of how FioranoMQ profile can be configured to store principal realms (users and groups) related to the FioranoMQ server. As an example, the view of how the users and groups are stored in the LDAP provider is extracted using the Apache Directory Studio.

LDAP for Principal store in FioranoMQ can be configured in the following way. After opening the profile in Fiorano Studio for offline editing and changing the Principal Manager implementation to LDAP as given in Section 7.5, changes have to be made at the following node –

Fiorano > security > PrincipalManager > LdapPrincipalManager



Properties of LdapPrincipalManager

ObjectName	Fiorano.security.PrincipalManager:Service...
InstanceOf	PrincipalManager
Description	Principal Manager uses LDAP storage for u...
Component Instance Configuration	
LdapProviderUrl	**ldap://192.168.1.143:389/**
LdapProviderDn	o=fiorano,c=US
Principal	cn=FMQRoot,o=fiorano,c=US
Credentials	secret
LdapInitialCtxFactory	com.sun.jndi.ldap.LdapCtxFactory
LdapSecurityAuthentication	simple
LdapPollInterval	-1
LdapPrimaryServerReconnectAttempts	5
BackupLdapProviderUrl	
BackupLdapProviderDn	null
BackupPrincipal	
BackupCredentials	secret
BackupLdapInitialCtxFactory	
BackupLdapSecurityAuthentication	simple
LdapUserClassTop	top
LdapUserClassPerson	Person
LdapUserClassOrganizationalPerson	OrganizationalPerson
LdapUserClassInetorgPerson	inetOrgPerson
LdapGroupClassTop	top
LdapGroupClassUniqueNames	groupofuniquenames
LdapGroupClassUniqueMember	o=fiorano,c=US
LdapUserNameAttribute	uid
LdapUserPasswordAttribute	************
LdapGroupNameAttribute	cn
LdapGroupUserNameAttribute	uniquemember
LdapUserDn	**cn=FioranoMQUsers,o=fmq**
LdapGroupDn	**cn=FioranoMQGroups,o=fmq**

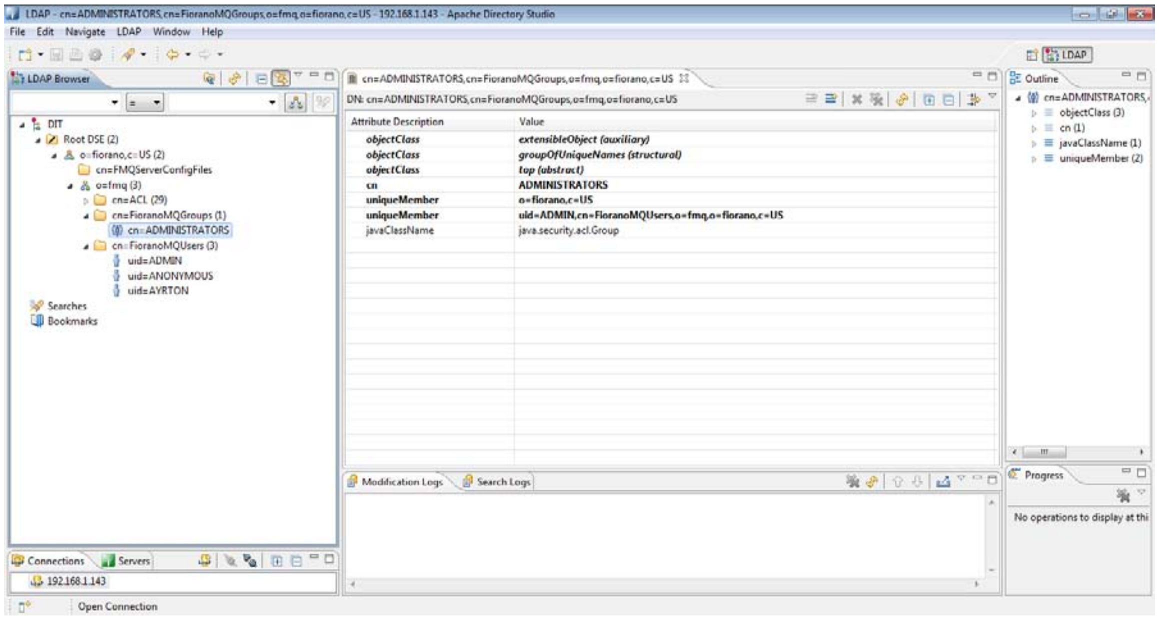
LdapPrincipalManager
ServiceProvider instance part of the profile

FioranoMQ saved.

Figure – LDAP_Conf 1

For more information on the parameters given in the above picture, refer to the FioranoMQ parameter reference guide which can be downloaded from the location http://www.fiorano.com/devzone/documentation.php.

Once the FioranoMQ is configured to use LDAP to store users and groups and the server is started, it sequentially creates them. The way in which the users and groups are stored in the LDAP-provider is illustrated using the following figure.



LDAP - cn=ADMINISTRATORS,cn=FioranoMQGroups,o=fmq,o=fiorano,c=US - 192.168.1.143 - Apache Directory Studio

DN: cn=ADMINISTRATORS,cn=FioranoMQGroups,o=fmq,o=fiorano,c=US

Attribute Description	Value
objectClass	***extensibleObject (auxiliary)***
objectClass	***groupOfUniqueNames (structural)***
objectClass	***top (abstract)***
cn	**ADMINISTRATORS**
uniqueMember	**o=fiorano,c=US**
uniqueMember	**uid=ADMIN,cn=FioranoMQUsers,o=fmq,o=fiorano,c=US**
javaClassName	java.security.acl.Group

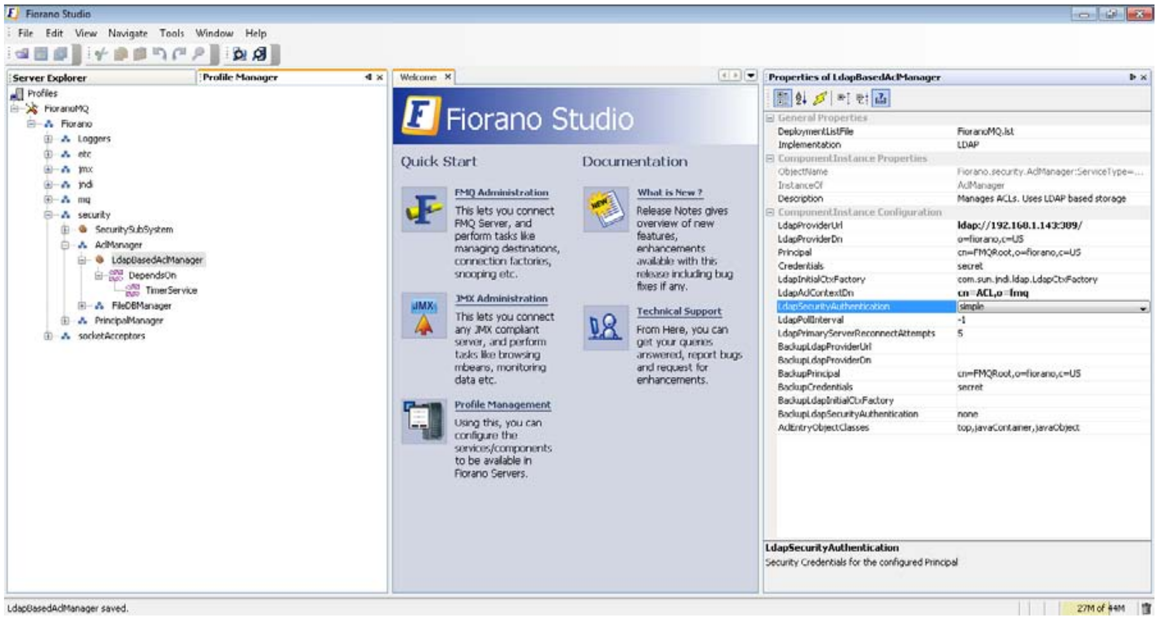
Figure – LDAP_Browser 1

7.9.3.2 Configuration for Access Control Lists (ACLs)

Here is an example of how FioranoMQ profile can be configured to store Access Control Lists (ACLs) related to the FioranoMQ Admin Objects like Queues, Topics, Connection Factories and other ACLs related to Lookup, AdminConnection etc. in the LDAP-provider. As an example, the view of how the ACLs are stored in the LDAP provider is extracted using the Apache Directory Studio.

LDAP for ACL store in FioranoMQ can be configured in the following way. After opening the profile in Fiorano Studio for offline editing and changing the ACL implementation to LDAP as given in Section 7.4, changes have to be made at the following node –

Fiorano > security > AclManager > LdapBasedAclManager



This lets you connect FMQ Server, and perform tasks like managing destinations, connection factories, snooping etc.

This lets you connect any JMX compliant server, and perform tasks like browsing mbeans, monitoring data etc.

Using this, you can configure the services/components to be available in Fiorano Servers.

Release Notes gives overview of new features, enhancements available with this release including bug fixes if any.

From Here, you can get your queries answered, report bugs and request for enhancements.

Properties of LdapBasedAclManager

General Properties	
DeploymentListFile	FioranoMQ.lst
Implementation	LDAP
ComponentInstance Properties	
ObjectName	Fiorano.security.AclManager:ServiceType=...
InstanceOf	AclManager
Description	Manages ACLs. Uses LDAP based storage
ComponentInstance Configuration	
LdapProviderUrl	**ldap://192.168.1.143:389/**
LdapProviderDn	o=fiorano,c=US
Principal	cn=FMQRoot,o=fiorano,c=US
Credentials	secret
LdapInitialCtxFactory	com.sun.jndi.ldap.LdapCtxFactory
LdapAclContextDn	**cn=ACL,o=fmq**
LdapSecurityAuthentication	simple
LdapPollInterval	-1
LdapPrimaryServerReconnectAttempts	5
BackupLdapProviderUrl	
BackupLdapProviderDn	
BackupPrincipal	cn=FMQRoot,o=fiorano,c=US
BackupCredentials	secret
BackupLdapInitialCtxFactory	
BackupLdapSecurityAuthentication	none
AclEntryObjectClasses	top,javaContainer,javaObject

LdapSecurityAuthentication
Security Credentials for the configured Principal

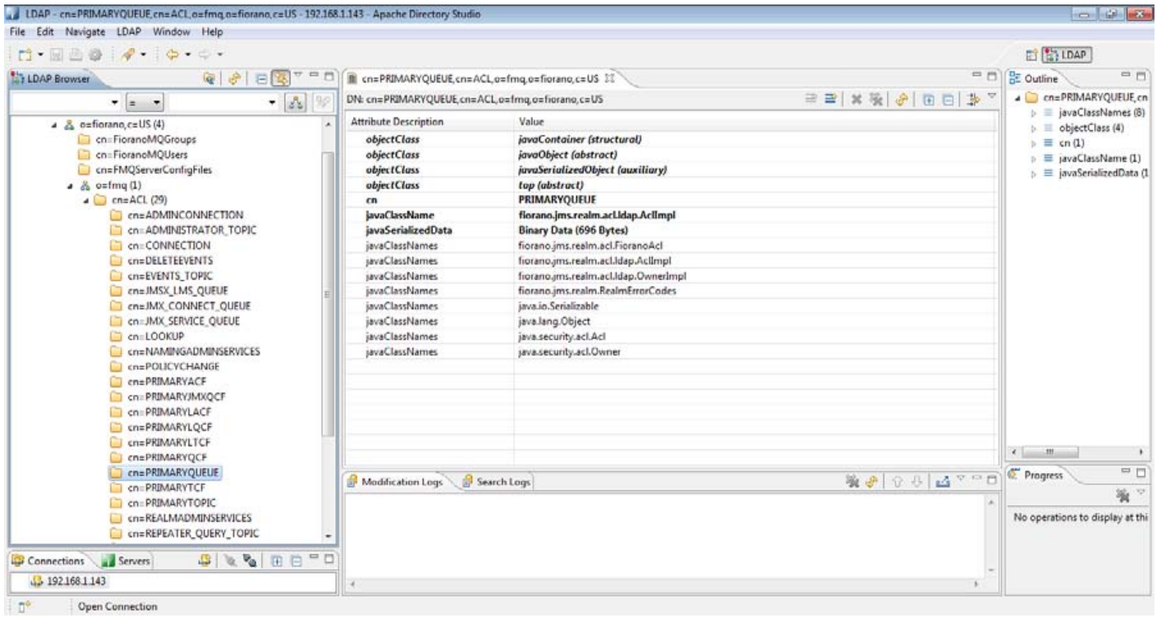
LdapBasedAclManager saved.

27M of 44M

Figure – LDAP_Conf2.png goes here

For more information on the parameters given in the above picture, refer to the FioranoMQ parameter reference guide which can be downloaded from the location http://www.fiorano.com/devzone/documentation.php.

Once the FioranoMQ is configured to use LDAP to store Access Control Lists (ACLs) and the server is started, it sequentially creates the ACLs for each of the destinations. The way in which the ACLs are stored in the LDAP-provider is illustrated using the following figure.


DN: cn=PRIMARYQUEUE,cn=ACL,o=fmq,o=fiorano,c=US

Attribute Description	Value
objectClass	*javaContainer (structural)*
objectClass	*javaObject (abstract)*
objectClass	*javaSerializedObject (auxiliary)*
objectClass	*top (abstract)*
cn	**PRIMARYQUEUE**
javaClassName	**fiorano.jms.realm.acl.ldap.AclImpl**
javaSerializedData	**Binary Data (696 Bytes)**
javaClassNames	fiorano.jms.realm.acl.FioranoAcl
javaClassNames	fiorano.jms.realm.acl.ldap.AclImpl
javaClassNames	fiorano.jms.realm.acl.ldap.OwnerImpl
javaClassNames	fiorano.jms.realm.RealmErrorCodes
javaClassNames	java.io.Serializable
javaClassNames	java.lang.Object
javaClassNames	java.security.acl.Acl
javaClassNames	java.security.acl.Owner

Figure – LDAP_Browser 2

7.10 XML Security Realm

1. Open the desired profile for off-line editing through the **Profile Manager** using Studio, as explained in section 4.8.1 How to Enable Pinging
2. Modify the Implementation property of the Principal Manager and the ACL Manager to XML. For more information on how to modify the Principal Manager and the ACL Manager, refer to the sections Modifying ACL Manager Implementation and Modifying Principal Manager Implementation .
3. Configure **Principal Manager** and **ACL Manager**.
4. Right-click on the FioranoMQ node and select **Save** from the pop-up menu.

7.10.1 Configuring Principal Manager and ACL Manager

7.10.1.1 Principal Manager

UserFileName

The name and path of the xml file containing 'User' information. The default file is user.xml.

GroupFileName

The name and path of the xml file containing 'Group' information. The default file is group.xml.

Path

This is the absolute or the relative path where User and Group files are stored. User and User Group files are saved to the location specified in the absolute path, whereas specifying a relative path saves User and User Group files to [FMQ_DB_PATH]\[relative path entered]. The default relative path is %FIORANO_HOME%\fmq\profiles\%selectedProfile%\run\realm\principal.

7.10.1.2 ACL Manager

FileName

The name and path of the xml file containing 'User' information. The default file is acl.xml.

MaxAcePerACL

Maximum number of entries that an ACL can store. The default number is 100.

Path

This is the absolute or the relative path where xml files are stored. User and User Group files are saved to the location specified in the absolute path, whereas specifying a relative path saves User and User Group files to [FMQ_DB_PATH]\[relative path entered]. The default relative path is %FIORANO_HOME%\fmq\profiles\%selectedProfile%\run\realm\principal.

7.10.2 Sample xml files

7.10.7.1 User.xml

```
<?xml version="1.0"?>

<UserManager>
<User>
     <Name>ADMIN</Name>
     <Password></Password>
     </User>
    <User>
    <Name>Anonymous</Name>
    <Password></Password>
    </User>
    </UserManager>
```

Here:

- **<UserManager>** Is the root element of the UserManager.
- **<User>** The UserManager may consist of one or more users.
- **<Name>** The name of the user. This is used to identify the user entry and is used in the ACLS and Groups.
- **<Password>** Is the password of the user. This is stored in encrypted form. It, therefore, cannot be specified from outside the system.

7.10.7.2 Group.xml

```
<?xml version="1.0"/>

<GroupManager>
<Group>
                <Name>EVERYONE</Name>
                </Group>
                </GroupManager>
```

Where:

- **<GroupManager>** Root element of the GroupManager.
- **<Group>** The GroupManager may consist of one or more groups.
- **<Name>** This is the name of the group. It is used to identify the group entry and is used in the ACL table.
- **<Member>** A group can consist of one or more members. These members must exist in the user table.

7.10.7.3 acl.xml

```
</AclManager>
       <ACL>
                </Name>LOOKUP</Name>
                <AclEntry Type="POS">
                <Principal>EVERYONE</Principal>
                <Permission>LOOKUP</Permission>
                </AclEntry>
                </ACL>
                </AclManager>
```

Where:

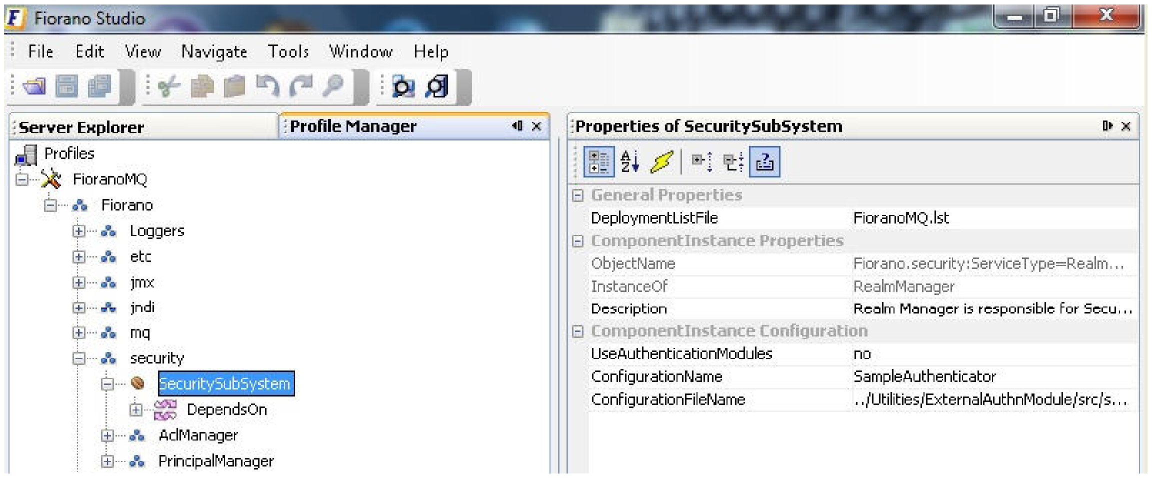
- **<AclManager>** Root element of the ACL dtd.
- **<ACL>** The ACL Manager consists of one or more ACLs and holds information about all the ACLs. **<Name>** Specifies the name of the ACL.
- **<AclEntry>** An ACL consists of one or more ACL Entries, which may be either negative (NEG) or positive (POS).
- **<Principal>** An ACL Entry consists of a Principal, which can be a User or a User Group.
- **<Permission>** An ACL Entry consists of 0 permissions or 1 permission to perform a task.

7.11 Plug-in Based Authentication Support

7.11.1 Enabling Plug-in Based User Authentication in Server

Following are the steps to enable plug-in based authentication in FioranoMQ Server-

1. Launch the admin studio and open the profile (by default FioranoMQ) in offline mode.
2. Navigate to FioranoMQ -> Fiorano -> security -> SecuritySubSystem. In the properties pane, set UseAuthenticationModules property to 'yes'.
3. Save the configuration.



General Properties	
DeploymentListFile	FioranoMQ.lst
ComponentInstance Properties	
ObjectName	Fiorano.security:ServiceType=Realm...
InstanceOf	RealmManager
Description	Realm Manager is responsible for Secu...
ComponentInstance Configuration	
UseAuthenticationModules	no
ConfigurationName	SampleAuthenticator
ConfigurationFileName	../Utilities/ExternalAuthnModule/src/s...

7.11.2 Using Authentication Modules to Authenticate a User

Using JAAS modules, FioranoMQ server is able integrate with an external pluggable security service provider like a local UNIX/Linux operating system and for LDAP based authentication which can store the user login information. The implementation w.r.t. interaction with this external security service provider (like creating the JDBC connection, creating the SSL-enabled LDAP connection and querying the RDBMS/LDAP-provider to validate the user authentication information etc.,) and thereby getting the required authentication information (PASSED? FAILED?) is to be done externally using the JAAS modules and FioranoMQ calls these APIs to get the required authentication information (PASSED? FAILED?) and based on it authenticate the user for performing one of the above operations. This section will demonstrate how the necessary plug-in modules need to be provided for the authentication purpose.

7.11.2.1 Login Configuration

JAAS authentication is performed in a pluggable fashion. This permits Java applications (in this case FioranoMQ Server) to remain independent from underlying authentication technologies. New or updated technologies can be plugged in without requiring modifications to the application itself. An implementation for a particular authentication technology to be used via LoginModule(s) is determined at runtime. The implementation is specified in a login configuration file.

The configuration file to be used can be specified in one of two ways:

1. Through server configuration (preferred)
    a. Launch the admin studio and open the profile (by default FioranoMQ) in offline mode.
    b. Navigate to FioranoMQ -> Fiorano -> security -> SecuritySubSystem. In the properties pane, set ConfigurationFileName property to 'the location of a desired configuration file'.
    c. Save the configuration.
2. Server start-up parameters
    a. Open %FIORANO_HOME%/fmq/bin/fmq.conf file and under <java.system.props> tag, add the following line:
        i. java.security.auth.login.config=%location of a desired configuration file%

As a login configuration file can consist of one or more entries, each specifying which underlying authentication technology should be used for a particular application or applications, the particular configuration entry to be used by the FioranoMQ server is specified as follows:

1. Launch the admin studio and open the profile (by default FioranoMQ) in offline mode.
2. Navigate to FioranoMQ ->Fiorano ->security ->SecuritySubSystem. In the properties pane, set ConfigurationName property to 'desired configuration name'.
3. Save the configuration.

For more information as to what a login configuration file is, what it contains, see online documentation for JAAS.

7.11.2.2 LoginModule

LoginModule describes the interface implemented by authentication technology providers (system administrators). LoginModules are plugged in under applications to provide a particular type of authentication. While FioranoMQ invokes the LoginContext API, authentication technology providers should implement the LoginModule interface. As mentioned in previous section, the Configuration specifies the LoginModule(s) to be used with a particular login application i.e., FioranoMQ server. Therefore different LoginModules can be plugged in seamlessly under the FioranoMQ server without any server-side configurations. A sample LoginModule must implement the following methods of javax.security.auth.spi.LoginModule

- boolean abort()
- boolean commit()
- void initialize(Subject subject, CallbackHandler callbackHandler, Map sharedState, Map options)
- boolean login()
- boolean logout()

Once the necessary implementations of the LoginModule interface are done, they should be included in the class path of the FioranoMQ Server such that when the server invokes the LoginContext API while authenticating the user credentials, the LoginModules (as per configuration) and inherently invoked successfully. This should be done as follows:

1. Locate the jar file which contains the class files of the implementation classes of LoginModule.
2. Open %FIORANO_HOME%/fmq/bin/fmq.conf file and under “3RD PARTY LIBRARIES” as the location to the above jar file.

Example implementation

Detailed explanation and an example implementation of LoginModule com.fiorano.jms.auth.SampleLoginModule is provided in %FIORANO_HOME%/fmq/Utilities/ExternalAuthnModule directory. More details of the class and other dependencies are present the readme file in the same directory.

Chapter 8: Large Message Support

For information about configuring profiles through text-based files, refer to FioranoMQ Getting Started.

8.1 Using FioranoMQ LMS APIs

8.1.1 Interface ILargeMessage

Purpose

The ILargeMessage interface provides APIs with the ability to transfer large files. It also provides APIs with the ability to check the status of a file transfer and to resume a file transfer.

Method Summary

public LMTransferStatus getMessageStatus()

Retrieves the status of the message. 'Status' refers to the number of bytes transferred, the number of bytes left to be transferred, and so on.

public void setLMStatusListener(LMStatusListener listener, int updateFrequency);

Sets the status listener for the message. This API detects the status of a message being transferred, asynchronously.

public LMStatusListener getLMStatusListener();

Retrieves the status listener for the message.

public void saveTo(String fileName, boolean isBlocking) throws FioranoException;

Saves the contents of the message in the file specified.

public void resumeSaveTo(boolean isBlocking) throws FioranoException;

Resumes an incomplete transfer. This API resumes the process of saving the contents of a message in the file specified.

public void resumeSend() throws FioranoException;

Resumes an incomplete transfer. This API resumes the process of sending a message. It is used when this process could not be completed due to an internal error or due to a problem originating at the client's side.

public void cancelAllTransfers() throws FioranoException;

Cancel all messages transfer in process for transferring this large message. Cancelling a transfer removes the 'resume' information related to that transfer. A transfer once cancelled, cannot be resumed.

public void cancelTransfer(int consumerID) throws FioranoException;

> Cancels the transfer process of the message belonging to the consumer identified by the consumer ID. Every consumer has a unique consumer ID assigned by the producer when the transfer starts. The API can be used by the sender and by the receiver.
>
> Canceling a transfer removes the 'resume' information related to that transfer. A transfer, once cancelled, cannot be resumed.

public void suspendAllTransfers() throws FioranoException;

> Suspends all message transfers transferring this large message, temporarily. Suspending a transfer only stops the thread that is performing the message transfer. No 'resume' related information is deleted. A transfer that is suspended can be resumed using resumeSend () and resumeSaveTo() APIs.

public void suspendTransfer(int consumerID) throws FioranoException;

> Suspends the message transfer specified by the consumer ID, temporarily. Suspending a transfer only stops the thread that is performing the message transfer. No 'resume' related information is deleted. Hence a suspended transfer can be resumed using resumeSend () and resumeSaveTo() APIs.

public void setFragmentSize(int size)

> Sets the fragment size of the message.

public int getFragmentSize()

> Retrieves the fragment size of the message.

public void setWindowSize(int size)

> Sets the window size for the message. Window size indicates the time interval after which the receiver sends an acknowledgement for message fragments received.

public int getWindowSize()

> Retrieves the window size of the message.

public void setRequestTimeoutInterval(long timeout)

> Sets the time duration for which the sender waits for large message requests sent by the receiver.

public long getRequestTimeoutInterval()

> Retrieves the duration for which the sender waits for the receiver's request.

public void setResponseTimeoutInterval(long timeout)

Sets the time duration for which the receiver waits for a message fragment sent by the sender.

public long getResponseTimeoutInterval()

Retrieves the time duration for which the receiver waits for a message fragment from the sender.

8.1.2 Interface ILMConnection

Purpose

ILMConnection provides APIs to retrieve messages that could not be entirely sent or received.

Method Summary

public void setResumeDirectory(java.lang.String dir)

Sets the User directory within which the resume file is saved

public String getResumeDirectory()

Retrieves the user directory in which the resume file is saved

public java.util.Enumeration getUnfinishedMessagesToSend ()

Retrieves the enumeration of ILargeMessages that could not be sent in entirety.

public java.util.Enumeration getUnfinishedMessagesToReceive ()

Retrieves the enumeration of ILargeMessages that fail to be transferred in their entirety.

public boolean hasTransfersInExecution ()

Indicates whether a connection has any ongoing transfers. This API is used by the application to close the connection depending upon whether or not the connection has transfers in execution.

8.1.3 Class LMTransferStatus

Purpose

Class LMTransferStatus provides the status of a message transfer. The 'status' of a message transfer refers to the number of bytes transferred, the number of bytes to be transferred, the last fragment of bytes transferred successfully, the percentage of progress of the transfer, and so on.

Constants Summary

public static final byte LM_TRANSFER_NOT_INIT=0x01;

Indicates that the transfer has not yet started.

public static final byte LM_TRANSFER_IN_PROGRESS=0x02;

Indicates that the transfer is currently in progress.

public static final byte LM_TRANSFER_DONE

Indicates that the transfer is complete.

public status LM_TRANSFER_ERR

Indicates that an error had occurred during the transfer.

Method Summary

public long getBytesTransferred()

Returns the number of bytes transferred.

public long getBytesToTransfer()

Returns the number of bytes left to be transferred.

public long getLastFragmentID()

Returns the fragment number of the last fragment sent successfully.

public float getPercentageProgress()

Returns the progress of the message transfer, as a percentage.

public byte getStatus()

Returns the status of the message transfer. The status of a message can be any one of the 'status' mentioned above.

public boolean isTransferComplete()

Returns a boolean value indicative of whether the transfer was completed successfully or not.

public ILargeMessage getLargeMessage ()

Returns the reference of a large message whose status is displayed by LMTransferStatus.

8.1.4 Interface LMStatusListener

Purpose

The LMStatusListener interface is used to detect, asynchronously, the status of a message being transferred.

Method Summary

public void onLMStatus(fiorano.jms.lm.LMTransferStatus status, FioranoException exception)

Method callback is invoked in the event of a message transfer status change or in the event of an exception that occurs during message transfer.

8.1.5 Class FioranoLMErrorCodes

Purpose

Class FioranoLMErrorCodes defines the error codes and error messages used by LMS.

LM_INVALID_SOURCE_FILE

This exception is encountered if the source file specified in the message is invalid.

LM_CSP_ENABLED

This exception is encountered if CSP/Durable connections are enabled when using LMS.

LM_REQUEST_TIMEOUT

This exception is encountered if a request for a large message is not received from any consumer in the time specified.

LM_DECODE_LMS_PROPERTIES_FAILURE

This exception is encountered if an error occurs while decoding LMS properties.

LM_ACK_NOT_RECEIVED

This exception is encountered if the producer does not receive any acknowledgement of message fragments from the consumer in the time specified.

LM_FRAGMENT_SEND_FAILURE

This exception is encountered if the producer is not able to send the message fragment to the consumer.

LM_ACK_PROCESS_FAILURE

This exception is encountered if the producer is not able to process the acknowledgement received from the consumer.

LM_MESSAGE_TRANSFER_ERROR

This exception is encountered if an error occurs on the producer's side during a message transfer.

LM_INITIALIZATION_ERROR

This exception is encountered if an error occurs while initializing the message transfer.

LM_REQUEST_PROCESS_FAILURE

This exception is encountered if an error occurs while processing the message request received from the consumer.

LM_SEND_AVAILABILITY_FAILURE

This exception is encountered if the producer is not able to send denoting availability of a message to the consumer. The ‘availability’ message is a message that is sent by the producer to the consumer if there are any unfinished messages to resume.

LM_READ_DATA_ERROR

This exception is encountered if an error occurs while reading data from the source file.

LM_RECEIVE_FRAGMENT_ERROR

This exception is encountered if an error occurs while receiving message fragments sent by the producer.

LM_RECEIVE_FRAGMENT_TIMEOUT

This exception is encountered if the consumer is not able to receive the message fragment within the period of the specified ‘timeout’.

LM_UNABLE_TO_SEND_FRAGMENT_ACK

This exception is encountered if the consumer is not able to the send an acknowledgement regarding a message fragment to the producer.

LM_UNABLE_TO_RESUME_SEND

This exception is encountered when the producer is not able to resume a message transfer.

LM_INVALID_TARGET_FILE

This exception is encountered if the target file specified by the consumer is not valid.

LM_UNABLE_TO_SAVE_TARGET_FILE

This exception is encountered if an error occurs while saving the target file due to the unavailability of free disk space.

LM_UNABLE_TO_RESUME_RECEIVE

This exception is encountered when the consumer is not able to resume the message transfer.

LM_WRITE_DATA_ERROR

This exception is encountered if an error occurs while writing data onto the target file.

LM_UNABLE_TO_SAVETO_FILE

This exception is encountered when saveTo() API is invoked for a message that is not considered a large message.

LM_TRANSFER_NOT_STARTED

This exception is encountered when cancellation of a transfer is invoked for a message that is not considered a large message.

8.2 LMS Samples

8.2.1 Sending a large message

```
//create the large message

TextMessage lmsg = session.createTextMessage();

lmsg.setStringProperty("JMSX_LM_PATH", "D:\\batch\\lms_samples\\send.zip");

//register status listener

((ILargeMessage) lmsg).setLMStatusListener(new TrackStatus());

//start the message transfer

msgProducer.send(lmsg);
```

8.2.2 Receiving a large message

```
//receive the normal JMS message containing a reference to the large message

ILargeMessage lmsg = (ILargeMessage) qReceiver.receive();

//register status listener

lmsg.setLMStatusListener(new TrackStatus());

//start the message transfer

lmsg.saveTo("received.zip");
```

8.2.3 Resuming a message transfer on the send side

```
Enumeration enum =

((ILMConnection)jmsConnection).getUnfinishedMessagesToSend ();

while(enum.hasMoreElements())

{

//get the reference for the large unfinished message

ILargeMessage lmsg = (ILargeMessage)enum.nextElement();

//register status listener

lmsg.setLMStatusListener(new TrackStatus());
//resume the transfer

lmsg.resumeSend(); ;

}
```

8.2.4 Resuming a message transfer on the receive side

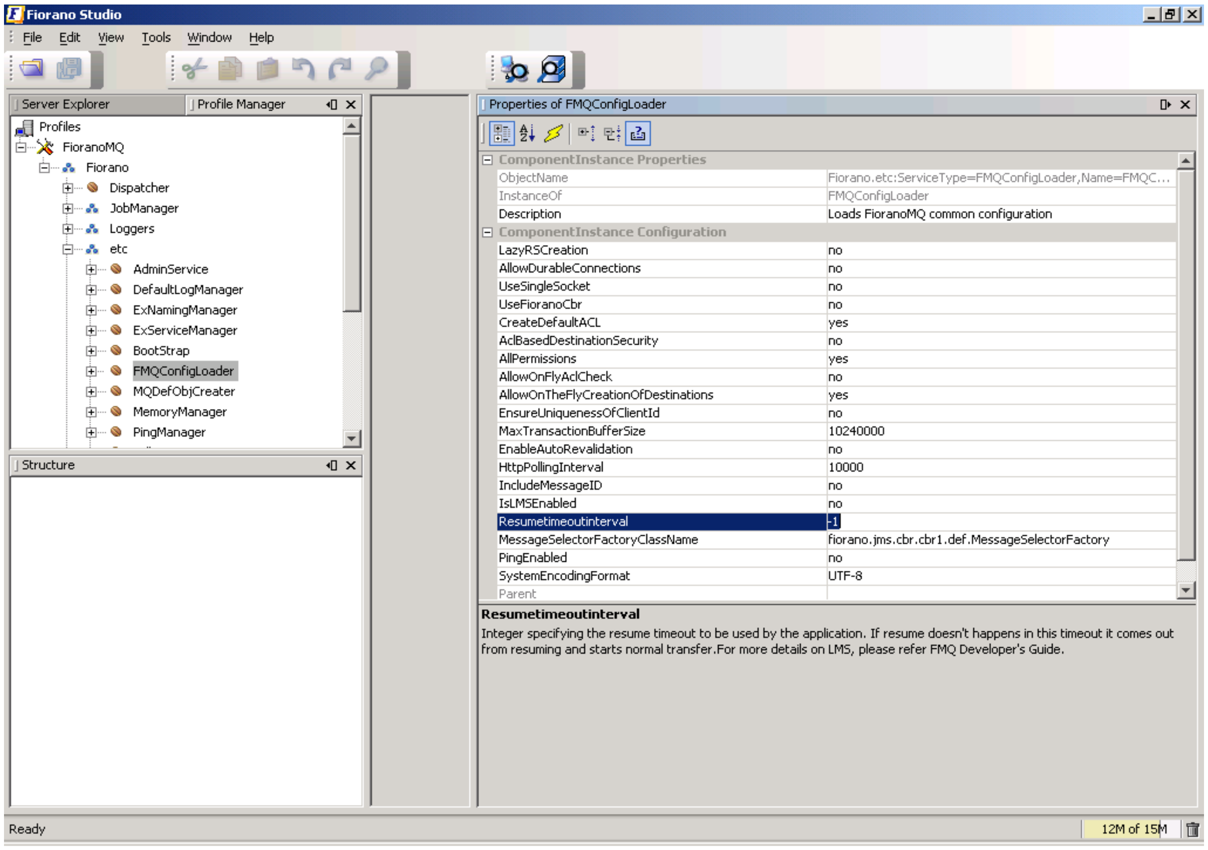
```
Enumeration enum =

((ILMConnection)jmsConnection).getUnfinishedMessagesToReceive ();

while(enum.hasMoreElements())

{

//recreate the large message

ILargeMessage lmsg = (ILargeMessage)umEnum.nextElement();

//registering status listener

lmsg.setLMStatusListener(new TrackStatus());

//resume the transfer

lmsg.resumeSaveTo ();

}
```

The time duration that the receiver waits till resuming the interrupted message transfer can be configured through the wait period in the parameter 'ResumeTimeoutInterval'. By default, the receiver waits for 15 seconds.

The resume timeout interval value can be set by following the steps below:

1. Launch **Admin Studio**. Launch **Fiorano Studio**. Open the **Profile Manager**. Right-click the **Profiles** node and select **Open Profile** from the pop-up menu. Select the desired profile for editing in the offline mode and click on the **Open** button.
2. In the **Profile Manager** pane navigate to **FioranoMQ > Fiorano >etc > FMQConfigLoader**. In the properties pane, set the value corresponding to the **Resumetimeoutinterval,** as shown in the figure below.

3. After making the above change, right-click the **FioranoMQ** domain in the **Profile Manager** and select the **Save** option from the pop-up menu.



ComponentInstance Properties	
ObjectName	Fiorano.etc:ServiceType=FMQConfigLoader,Name=FMQC…
InstanceOf	FMQConfigLoader
Description	Loads FioranoMQ common configuration
ComponentInstance Configuration	
LazyRSCreation	no
AllowDurableConnections	no
UseSingleSocket	no
UseFioranoCbr	no
CreateDefaultACL	yes
AclBasedDestinationSecurity	no
AllPermissions	yes
AllowOnFlyAclCheck	no
AllowOnTheFlyCreationOfDestinations	yes
EnsureUniquenessOfClientId	no
MaxTransactionBufferSize	10240000
EnableAutoRevalidation	no
HttpPollingInterval	10000
IncludeMessageID	no
IsLMSEnabled	no
Resumetimeoutinterval	-1
MessageSelectorFactoryClassName	fiorano.jms.cbr.cbr1.def.MessageSelectorFactory
PingEnabled	no
SystemEncodingFormat	UTF-8
Parent	

Resumetimeoutinterval
Integer specifying the resume timeout to be used by the application. If resume doesn't happens in this timeout it comes out from resuming and starts normal transfer.For more details on LMS, please refer FMQ Developer's Guide.

Chapter 9: HA

FioranoMQ Server within an Enterprise Server are required to have similar backend databases. This can be achieved by using a common or Shared database, or by setting up Replication between database instances. The first decision to be taken while setting up HA is the selection of one of these two options. The FioranoMQ installer comes with pre-built profiles for both Shared and Replication modes that are pre-configured to demonstrate HA on a single machine.

9.1 Default HA Profiles

This section of the document provides details about pre-configured HA profiles.

For information about configuring profiles through a text-based file, see *FioranoMQ Getting Started*.

These profiles can be found in the %FIORANO_HOME%\fmq\profiles\FioranoMQ_HA_rpl and FioranoMQ_HA_shared directories respectively. Primary and Secondary Servers are started by booting the container available in these profiles. The table below lists the profiles to use for the Primary and the Secondary Servers when setting up HA in Replication mode or Shared mode.

Mode	**Profile Directory**	**Meant For**
Replication	Fmq\profiles\FioranoMQ HA_rpl\HAPrimary	Primary Server
Replication	Fmq\profiles\FioranoMQ_HA_rpl\HASecondary	Secondary Server
Shared	fmq\profiles\FioranoMQ_HA_shared\HAPrimary	Primary Server
Shared	fmq\profiles\FioranoMQ_HA_shared\HASecondary	Secondary Server

Note: This document refers to the profile (in the FioranoMQ _HA_rpl or FioranoMQ_HA_shared directories) **as %SELECTED_HA_PROFILE%.** The possible values for **%SELECTED_HA_PROFILE%** are as shown in the table above.

9.2 Configuration Steps

Configuring HA in a FioranoMQ Server basically involves configuration of the modules below:

- HA Layer
- Admin Objects Store
- Security Store
- Message Store

The HA Layer configuration deals with the configuration of the Peer Server. The important configuration parameters defined in this layer are **isPrimary** (**TRUE** or **FALSE**), **HA Port**, **Backup server's IP Address**, **Backup server's Client Port,** and **Backup server's HA Port**. Configuration is required in both the Replication and the Shared HA mode. Additionally, Admin Objects Store, Security Store and Message Store must be configured for setting up Shared HA. When configuring HA, one of the following sets of instructions must be followed, depending on whether it is for Replication mode or for Shared mode.

Replication Mode

Step 1: Install FioranoMQ Servers

Step 2: HA Configuration

HA Shared Mode

Step 1: Install FioranoMQ Servers

Step 2: HA Configuration

Step 3: Admin Store Configuration

Step 4: Security Store Configuration

Step 5: Message Store Configuration

Configuring FioranoMQ HA in Replication mode is much easier, as the pre-created profiles are ready to be used; only the gateway IP needs to be modified. These profiles are pre-configured to demonstrate HA on a single machine. However, it is easy to setup HA on two different machines by configuring the HA Layer, as described below.

When setting up the server in Shared mode, in addition to setting up the HA Layer, the server configuration needs to be modified so as to point it to the Shared Admin Store, Security Store and Message Store. Instructions for these modifications can be found in Steps 3 to 5 in the sections below.

9.2.1 Step 1: FioranoMQ Server(s) Installation

The FioranoMQ Server(s) that forms the Enterprise Server can be installed on a single machine or on two separate machines, varying in hardware and/or software configuration. When setting up HA on a single machine, no changes are required; the default configuration (for primary and Secondary Server) is ready to be used. However, when setting up HA on different machines, the IP addresses of the Peer Servers have to be configured in the HA Layer. The following section provides step by step instructions for configuring the HA Layer.

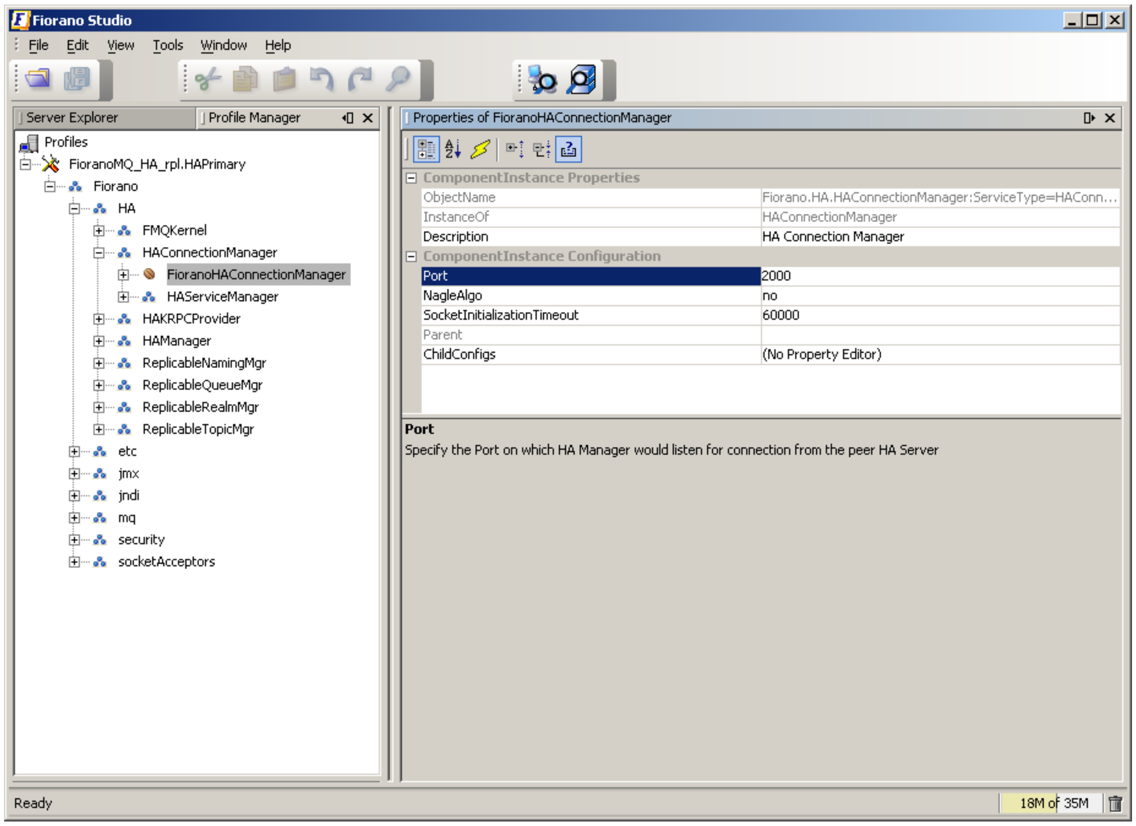
Note: The pre-created profile is configured to run HA on a single machine. Fiorano does not recommend setting up HA on just a single machine.

9.2.2 Step 2: HA Configuration

The HA Layer within a FioranoMQ server is responsible for creating a dedicated connection with the peer Backup Server. This dedicated connection is used for exchanging health and state information between peers being used. The following section provides details concerning the configuration of an HA Layer in a FioranoMQ Server and is applicable to both, Primary and Secondary Servers.

1. Launch Fiorano Studio, and click on the **Profile Manager** pane. Right–click the **Profiles** node, and select **Open Profile** from the pop-up menu. Select the required profile, and click on the **Open** button (%SELECTED_HA_PROFILE%) for offline editing, using the Profile Manager.
2. Configure self HA port: This port refers to the port number on which the HA Layer accepts a connection from its Peer Server. The HA port of the default HA profiles configured for the Primary Server is 2000 while that of the Secondary Server is 3000. In order to modify these values, navigate to the node mentioned below, and change the port number in the **PropertiesOf FioranoHAConnectionManager** pane.

%SELECTED_HA_PROFILE%-> Fiorano->HA->HAConnectionManager -> FioranoHAConnectionManager



Profiles
- FioranoMQ_HA_rpl.HAPrimary
  - Fiorano
    - HA
      - FMQKernel
      - HAConnectionManager
        - FioranoHAConnectionManager
        - HAServiceManager
      - HAKRPCProvider
      - HAManager
      - ReplicableNamingMgr
      - ReplicableQueueMgr
      - ReplicableRealmMgr
      - ReplicableTopicMgr
    - etc
    - jmx
    - jndi
    - mq
    - security
    - socketAcceptors

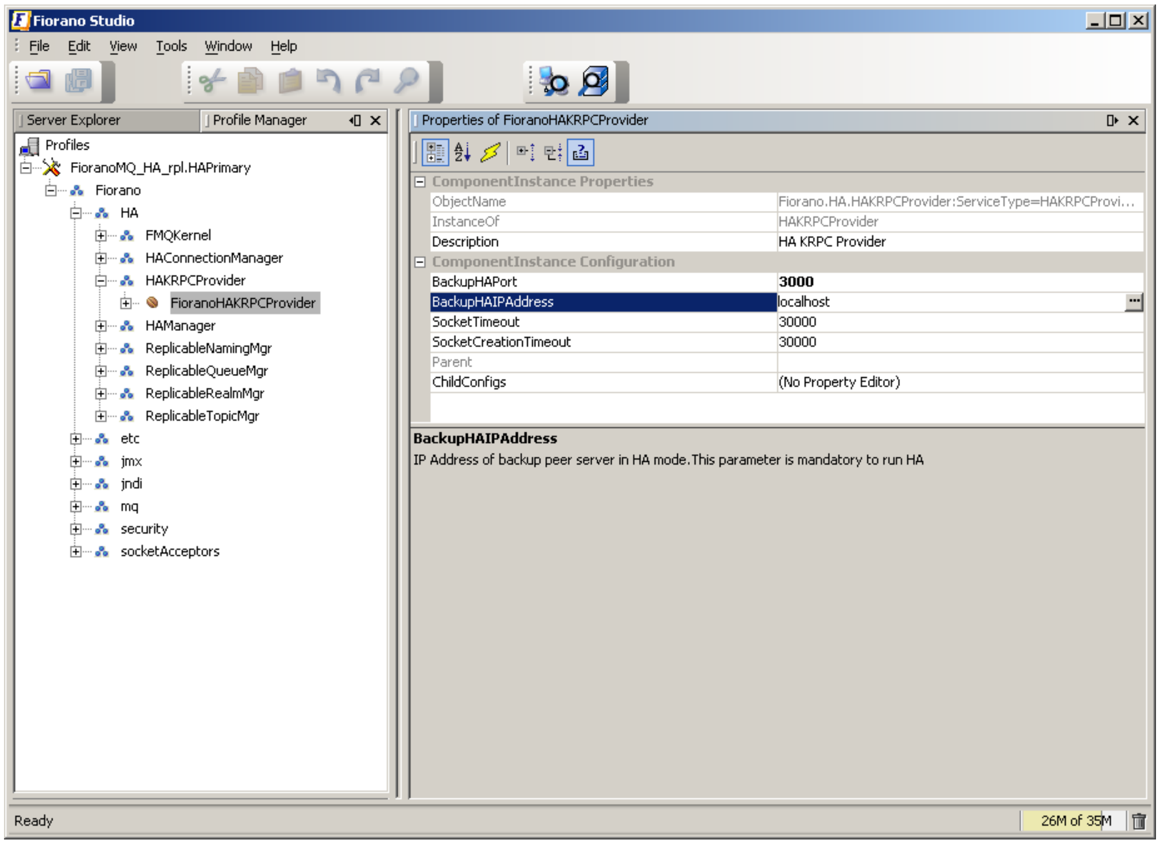
ComponentInstance Properties	
ObjectName	Fiorano.HA.HAConnectionManager:ServiceType=HAConn...
InstanceOf	HAConnectionManager
Description	HA Connection Manager
ComponentInstance Configuration	
Port	2000
NagleAlgo	no
SocketInitializationTimeout	60000
Parent	
ChildConfigs	(No Property Editor)

Port
Specify the Port on which HA Manager would listen for connection from the peer HA Server

Note: The screen-shot above shows the path for the **HAConnectionManager** node in Replication mode. The same path can be found when configuring the Connection Manager in Shared HA mode.

3. Configure the IP and HA port of the remote server: HA layer of a FioranoMQ Server connects to the Backup Server's HA layer. In order to configure the Backup Server's IP and its port, navigate to the node specified below, and set the values of *BackupHAIPAddress* and *BackupHAPort* displayed in the **Properties of FioranoHAKRPCProvider** pane.

%SELECTED_HA_PROFILE%-> Fiorano -> HA ->HAKRPCProvider -> FioranoHAKRPCObject



ComponentInstance Properties	
ObjectName	Fiorano.HA.HAKRPCProvider:ServiceType=HAKRPCProvi...
InstanceOf	HAKRPCProvider
Description	HA KRPC Provider
ComponentInstance Configuration	
BackupHAPort	3000
BackupHAIPAddress	localhost
SocketTimeout	30000
SocketCreationTimeout	30000
Parent	
ChildConfigs	(No Property Editor)

BackupHAIPAddress
IP Address of backup peer server in HA mode.This parameter is mandatory to run HA

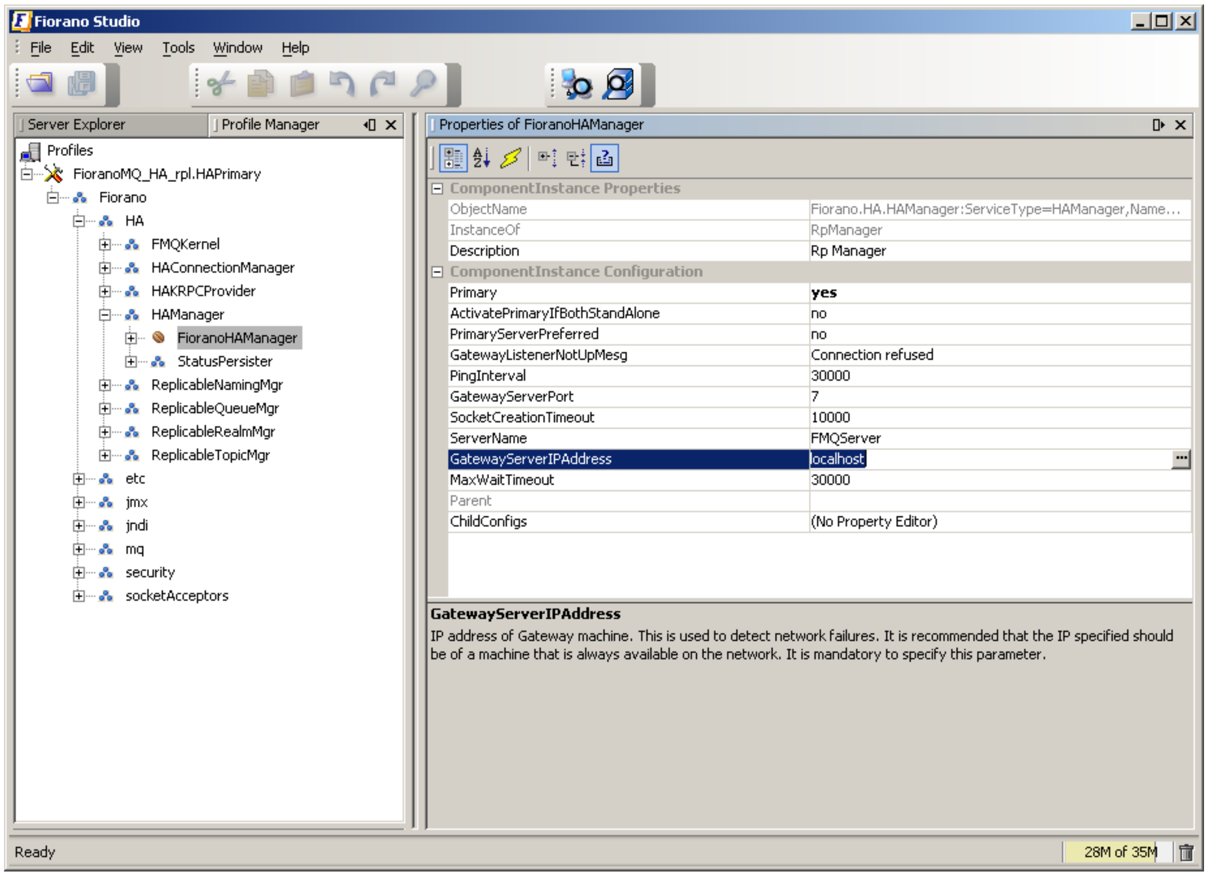
Note: The screen-shot above shows the path for the **HAKRPCProvider** node in the Replication mode. The same path can be found while configuring **HAKRPCProvider** in the Shared HA mode.

The backup HA IP address of the default profiles is configured to localhost. Their backup HA port is configured to 3000 in the Primary Server and to 2000 in the Secondary Server. In other words, by default, the Primary Server's HA layer tries to connect at localhost on port 3000, while the Secondary's HA Layer tries to connect to a localhost on port 2000.

The **BackupHA** port here does not refer to the port on which the Backup Server accepts client connections, but to the port on which it’s HA Layer listens. The value here should be the same as the value of **HAConnectionManager's** port, configured in the Peer Server.

4. Configure Gateway Server: HA Layer pings the gateway machine to determine the status of the network connectivity. The default profiles have Gateway configured to localhost and should be changed to a third independent reliable machine (preferably to the physical router machine on the network). In order to configure Gateway Server, navigate to the node specified below, and modify the value of **GatewayServerIPAddress** and **GatewayServerPort** in the **Propertiesof FioranoHAManager** pane.

%SELECTED_HA_PROFILE%-> Fiorano -> HA->FioranoHAManager



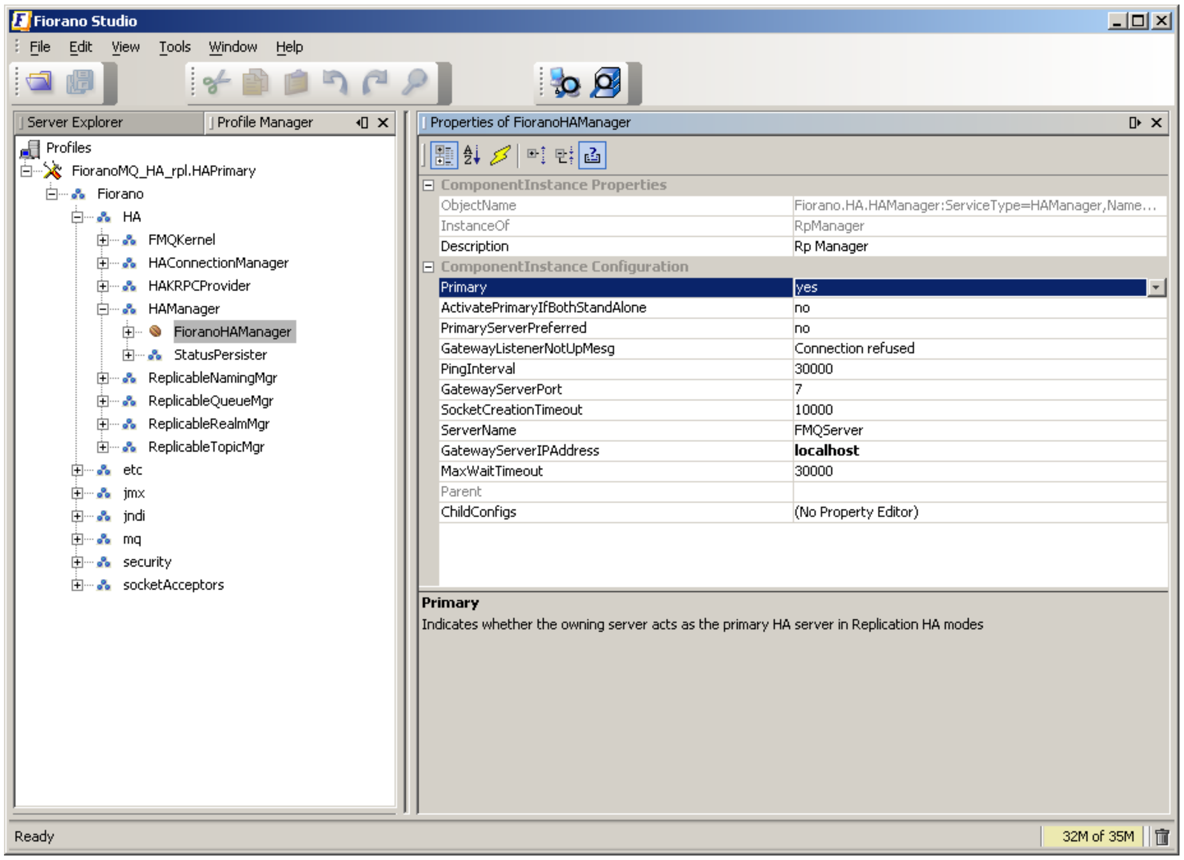
ComponentInstance Properties	
ObjectName	Fiorano.HA.HAManager:ServiceType=HAManager,Name...
InstanceOf	RpManager
Description	Rp Manager
ComponentInstance Configuration	
Primary	yes
ActivatePrimaryIfBothStandAlone	no
PrimaryServerPreferred	no
GatewayListenerNotUpMesg	Connection refused
PingInterval	30000
GatewayServerPort	7
SocketCreationTimeout	10000
ServerName	FMQServer
GatewayServerIPAddress	localhost
MaxWaitTimeout	30000
Parent	
ChildConfigs	(No Property Editor)

GatewayServerIPAddress
IP address of Gateway machine. This is used to detect network failures. It is recommended that the IP specified should be of a machine that is always available on the network. It is mandatory to specify this parameter.

Note: The screen-shot above shows the path for the **FioranoHAManager** node in the Replication mode. This path can be found when configuring **FioranoHAManager** in the Shared HA mode.

5. Configure **isPrimary**: Set **isPrimary** to **yes** for the Primary FioranoMQ Server and **no** for the Backup Server by modifying its value, as shown in the **properties** pane, by clicking on the node specified below:

%SELECTED_HA_PROFILE%-> Fiorano ->HA->HAManager -> FioranoHAManager



ComponentInstance Properties	
ObjectName	Fiorano.HA.HAManager:ServiceType=HAManager,Name...
InstanceOf	RpManager
Description	Rp Manager
ComponentInstance Configuration	
Primary	yes
ActivatePrimaryIfBothStandAlone	no
PrimaryServerPreferred	no
GatewayListenerNotUpMesg	Connection refused
PingInterval	30000
GatewayServerPort	7
SocketCreationTimeout	10000
ServerName	FMQServer
GatewayServerIPAddress	**localhost**
MaxWaitTimeout	30000
Parent	
ChildConfigs	(No Property Editor)

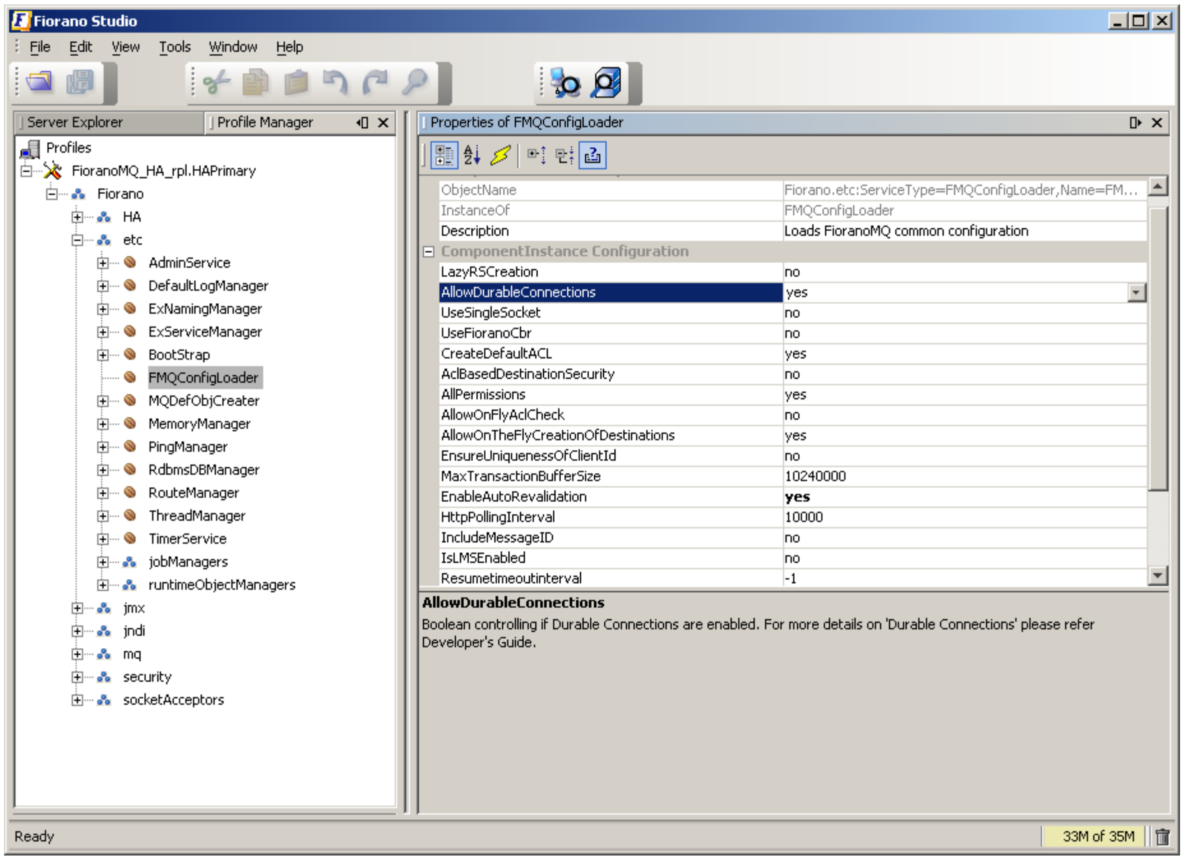
Primary
Indicates whether the owning server acts as the primary HA server in Replication HA modes

Note: The screen-shot above shows the path for the **FioranoHAManage**r node in the Replication mode. This path can be found when configuring **FioranoHAManager** in the Shared HA mode.

6. Common Configuration: HA requires enabling Durable Connections, Pinging and Auto Revalidation support in the FioranoMQ Server. This can be done by navigating to and modifying the parameters specified below, located in the properties pane.

%SELECTED_HA_PROFILE%-> Fiorano ->etc -> FMQConfigLoader

- Set AllowDurableConnections property to yes.
- Set PingEnabled to yes.
- Set EnableAutoRevalidation to yes.



```
Profiles
FioranoMQ_HA_rpl.HAPrimary
  Fiorano
    HA
    etc
      AdminService
      DefaultLogManager
      ExNamingManager
      ExServiceManager
      BootStrap
      FMQConfigLoader
      MQDefObjCreater
      MemoryManager
      PingManager
      RdbmsDBManager
      RouteManager
      ThreadManager
      TimerService
      jobManagers
      runtimeObjectManagers
    jmx
    jndi
    mq
    security
    socketAcceptors
```

Properties of FMQConfigLoader

ObjectName	Fiorano.etc:ServiceType=FMQConfigLoader,Name=FM...
InstanceOf	FMQConfigLoader
Description	Loads FioranoMQ common configuration
ComponentInstance Configuration	
LazyRSCreation	no
AllowDurableConnections	yes
UseSingleSocket	no
UseFioranoCbr	no
CreateDefaultACL	yes
AclBasedDestinationSecurity	no
AllPermissions	yes
AllowOnFlyAclCheck	no
AllowOnTheFlyCreationOfDestinations	yes
EnsureUniquenessOfClientId	no
MaxTransactionBufferSize	10240000
EnableAutoRevalidation	**yes**
HttpPollingInterval	10000
IncludeMessageID	no
IsLMSEnabled	no
Resumetimeoutinterval	-1

AllowDurableConnections
Boolean controlling if Durable Connections are enabled. For more details on 'Durable Connections' please refer Developer's Guide.

Note: The screen-shot above shows the path for **HAManager** node in the Replication mode. This path can be found when configuring **HAManager** in the Shared HA mode. Pre-created profiles are set to this configuration, and require no change when working with them.

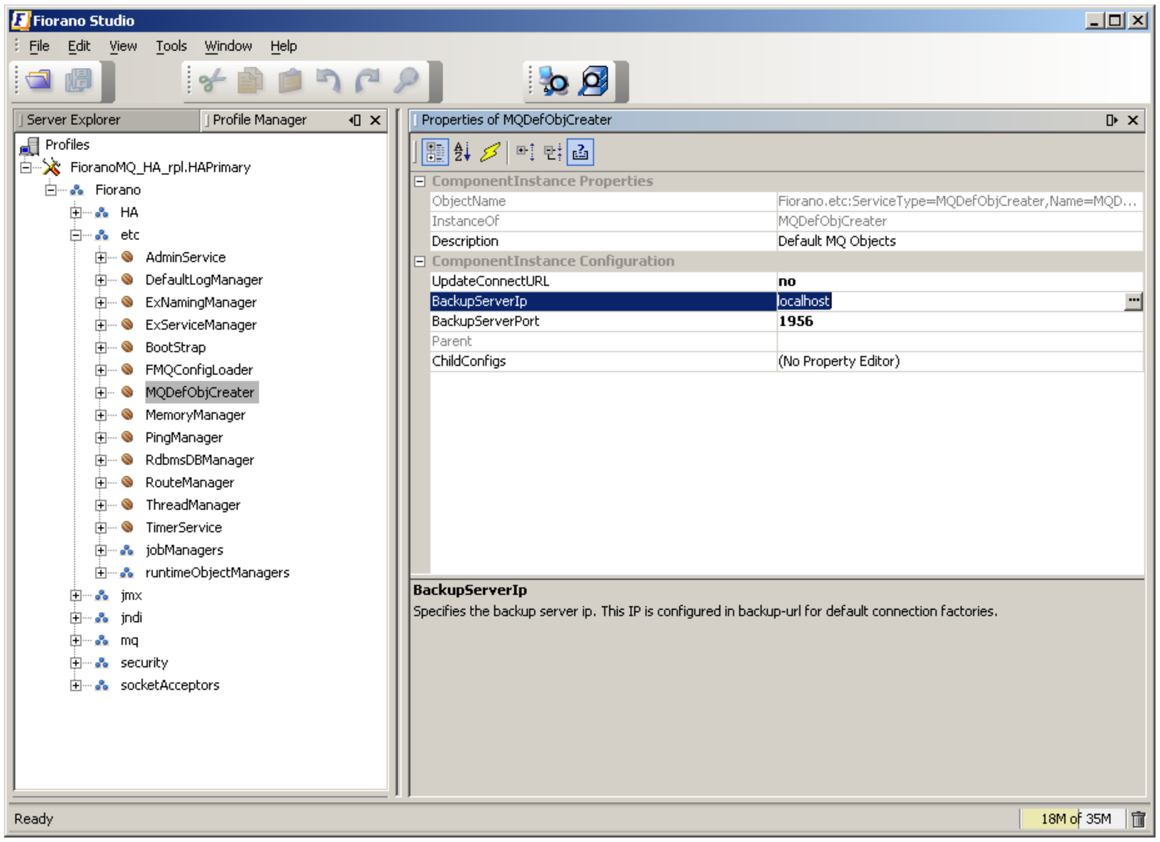
7. Configure the Backup Server's IP and Port: Specify the **BackUpServerIP** and **BackupServerPort** used by clients in the event of a connection failure. This IP Port should refer to the backup FioranoMQ Server's IP Address and to the Port on which it accepts client connections. This is done by navigating to the node specified below, and modifying the Backup Server's IP address and the Backup Server's port parameters within the **Properties** of the **MQDefObjCreator** pane.

%SELECTED_HA_PROFILE%->Fiorano->etc -> MQDefObjCreator

8. The FioranoMQ Server uses these parameters to set a Backup Server URL in the default connection factories. For client connections to fail-over, it is important that the Backup Server URL is properly configured. If the configuration is changed for the default connection factories, the database should be recreated, or a self-created connection factories database with correct URLs should be used.

   The IP address of the default profiles is configured for localhost, for Primary and Secondary Servers. The backup port is configured to 1956 in the Primary Server and to 1856 in the Secondary Server.

Note: In the default profile, the Secondary Server accepts client connections on port 1956, and the Primary Server accepts client connections on port 1856.



ComponentInstance Properties	
ObjectName	Fiorano.etc:ServiceType=MQDefObjCreater,Name=MQD...
InstanceOf	MQDefObjCreater
Description	Default MQ Objects
ComponentInstance Configuration	
UpdateConnectURL	no
BackupServerIp	localhost
BackupServerPort	1956
Parent	
ChildConfigs	(No Property Editor)

BackupServerIp
Specifies the backup server ip. This IP is configured in backup-url for default connection factories.

Note: The screenshot above shows the path for the **MQDefObjCreator** node in the Replication mode. This path can be found when configuring **HAManager** in the Shared HA mode.

9. Right-click on the **%SELECTED_HA_PROFILE% root**, and select the **Save** option from the pop-up menu to save this configuration.

Example Configuration

Parameter	**Primary Server**	**Backup Server**
URL	http://164.164.129.128:1856	http://164.164.129.108:1956
HAKRPCProvider		
BackupHAIPAddress	164.164.129.108	164.164.129.128
BackupHAPort	3000	2000
HAConnectionManager		
Port	2000	3000
Common Config		
Backup IP Address	164.164.129.108	164.164.129.128
Port	1956	1856
EnableDurableConnections	Yes	yes

Parameter	**Primary Server**	**Backup Server**
PingEnabled	Yes	yes
HAManager		
isPrimary	Yes	no
Gateway IP Address	164.164.129.225	164.164.129.225

10. After changing the HA Layer's configuration, clear the server's existing database prior to re-starting the server. The existing Data Store can be cleared by executing the *ClearDB* script in fmq/bin folder of the installation directory, with the profile name as an argument.

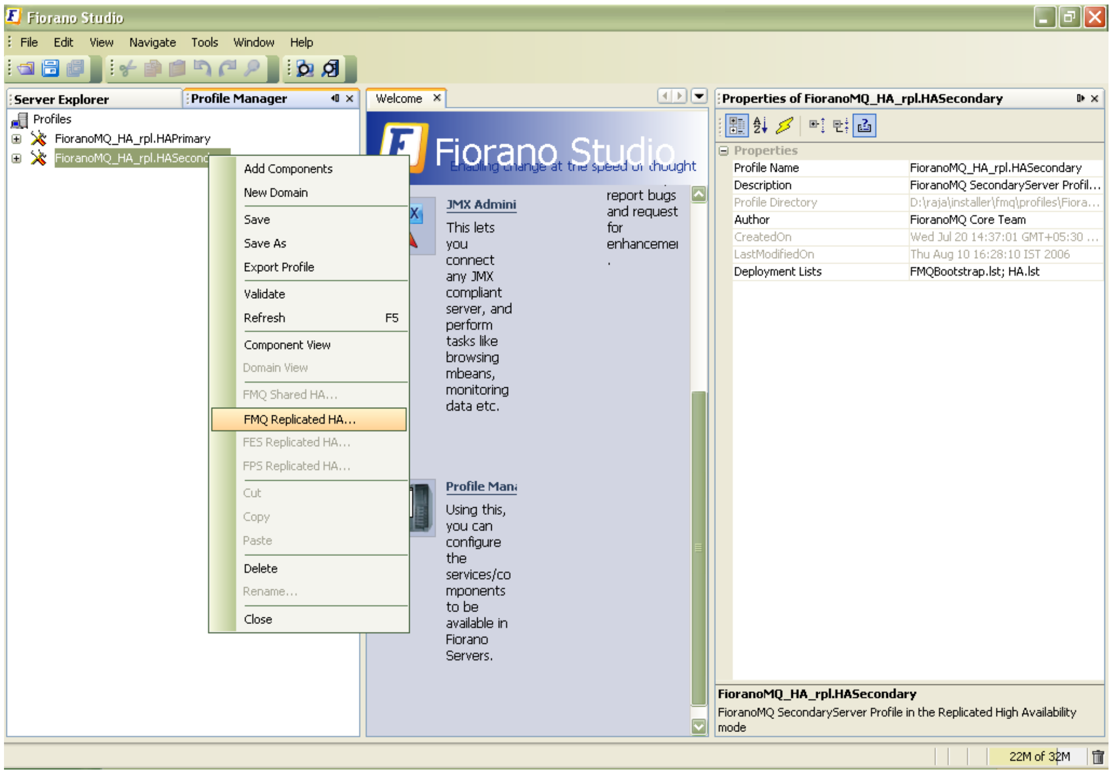
For example:

- clearDb FioranoMQ_HA_rpl/HAPrimary
- clearDb FioranoMQ_HA_shared/HAPrimary

These are required to re-create connection factories to include the backup of the IP address of the Peer Server. If clearing the entire Data Store is not possible, use only newly created connection factories for HA enabled applications.

9.2.2.1 Configuring FioranoMQ Replicated HA

1. To use Replicated HA, open **FioranoMQ_HA_rpl/HAPrimary**, and open the **FioranoMQ_HA_rpl/HASecondary** profile in the Studio **Profile Manager**. Right-click on the profile, and select **FMQ Replicated HA**. The **FMQ Replicated HA** wizard appears.



This lets you connect any JMX compliant server, and perform tasks like browsing mbeans, monitoring data etc.

report bugs and request for enhanceme

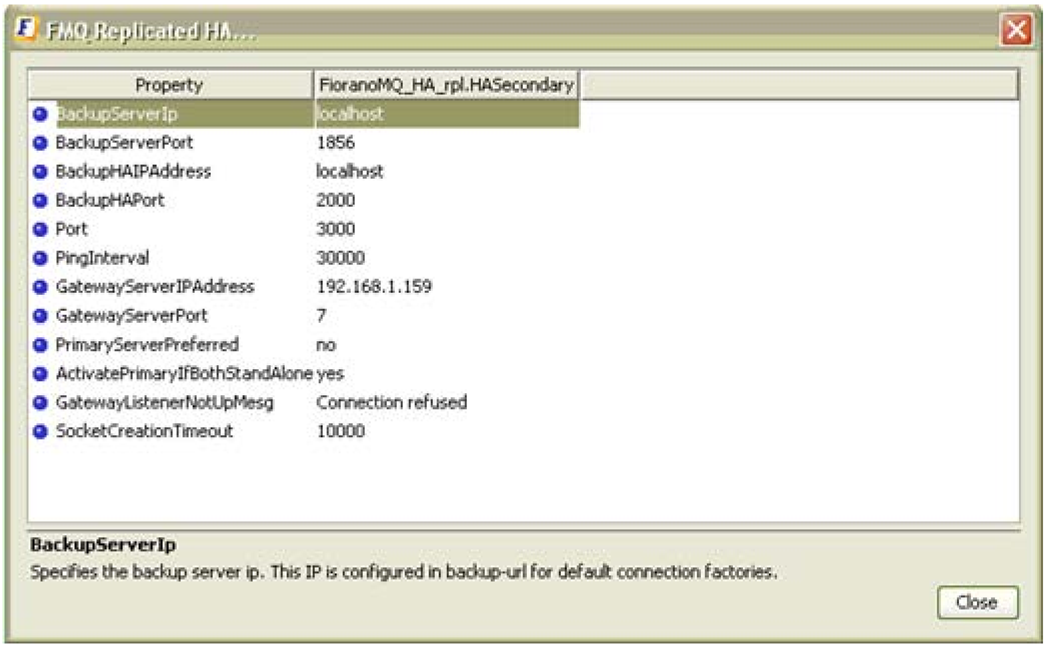
Using this, you can configure the services/components to be available in Fiorano Servers.

Properties of FioranoMQ_HA_rpl.HASecondary

Properties	
Profile Name	FioranoMQ_HA_rpl.HASecondary
Description	FioranoMQ SecondaryServer Profil...
Profile Directory	D:\raja\installer\fmq\profiles\Fiora...
Author	FioranoMQ Core Team
CreatedOn	Wed Jul 20 14:37:01 GMT+05:30 ...
LastModifiedOn	Thu Aug 10 16:28:10 IST 2006
Deployment Lists	FMQBootstrap.lst; HA.lst

FioranoMQ_HA_rpl.HASecondary
FioranoMQ SecondaryServer Profile in the Replicated High Availability mode

2. All required values can be configured using the wizard. After configuration, right-click on the node, and select the **Save** option from the pop-up menu.



Property	FioranoMQ_HA_rpl.HASecondary
BackupServerIp	localhost
BackupServerPort	1856
BackupHAIPAddress	localhost
BackupHAPort	2000
Port	3000
PingInterval	30000
GatewayServerIPAddress	192.168.1.159
GatewayServerPort	7
PrimaryServerPreferred	no
ActivatePrimaryIfBothStandAlone	yes
GatewayListenerNotUpMesg	Connection refused
SocketCreationTimeout	10000

BackupServerIp
Specifies the backup server ip. This IP is configured in backup-url for default connection factories.

The properties listed below are available in the **FMQ Replicated HA** wizard.

BackupServerIp

This IP represents the URL of FioranoMQ Backup Server. In the event of server failure, the server clients connect to the Backup Server.

BackupServerPort

This port represents the Backup Server port on which the Backup FioranoMQ Server runs.

BackupHAIPAddress

This IP represents the URL where another FioranoMQ HA Server runs. In the event of a failure of this HA Server, all the required synchronized data is available in the Backup HA Server. Here the HA Layer connects to the Backup HA Server.

BackupHAPort

This port represents the backup machine HA port on which another FioranoMQ Server can run.

Port

This port represents the FioranoMQ Server's HA port on which the HA Server runs.

PingInterval

This interval represents the time period, after which, the HA Server pings the HA Backup Server to determine if the server is running properly.

GatewayServerIPAddress

This value represents the IP address of the third machine present in the network. This value must not represent the machine on which the FioranoMQ Server runs.

GatewayServerPort

This value represents the port to which the HA Server pings. This port can be used only for this function.

ActiveLockAcquisitionInterval:

This parameter indicates the wait interval between each attempt to acquire the lock for Active Server. This value should be in multiples of half of the **PingInterval** value (**PingInterval** / 2). Otherwise, the server can try to acquire the lock on the next multiple of pingInterval/2.

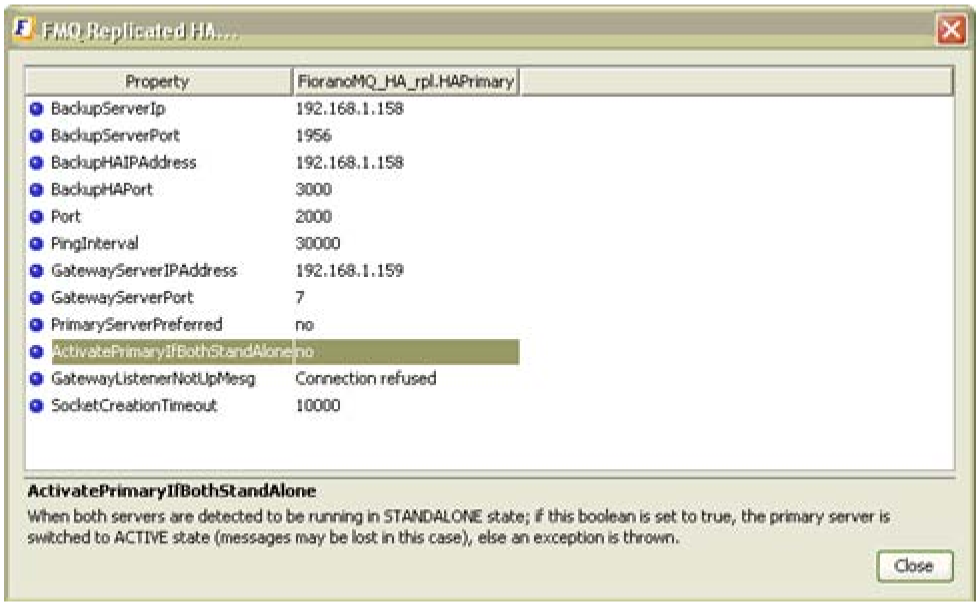
PassiveLockAcquisitionInterval:

This parameter indicates the wait interval for the Passive Server to acquire the lock when the link between active and Passive Server is down. This value should be greater than twice the value of the **ActiveLockAcquisitionInterval**, i.e.

2 * **ActiveLockAcquisitionInterval**, or an exception will be thrown. The server will not start if this value is not set properly.

All IP addresses and ports should represent the correct values. Even without the modification of any of the values, the HA Primary Server and the HA Secondary Server can run on a single machine.

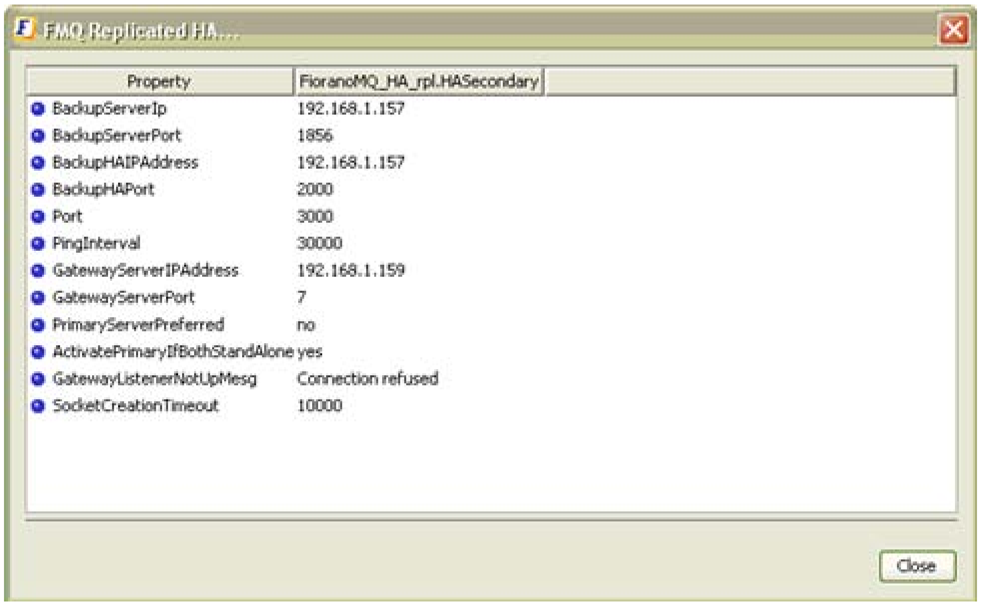
For example, if the HA Primary Server is running on a machine with IP Address 192.169.1.157., and the HA Secondary Server, on a machine with IP Address 192.169.1.159., the Gateway server address is given as 192.169.1.159. After configuration, the HA Primary Server the wizard should display the values, as shown in the figure below:



Property	FioranoMQ_HA_rpl.HAPrimary
BackupServerIp	192.168.1.158
BackupServerPort	1956
BackupHAIPAddress	192.168.1.158
BackupHAPort	3000
Port	2000
PingInterval	30000
GatewayServerIPAddress	192.168.1.159
GatewayServerPort	7
PrimaryServerPreferred	no
ActivatePrimaryIfBothStandAlone	no
GatewayListenerNotUpMesg	Connection refused
SocketCreationTimeout	10000

ActivatePrimaryIfBothStandAlone
When both servers are detected to be running in STANDALONE state; if this boolean is set to true, the primary server is switched to ACTIVE state (messages may be lost in this case), else an exception is thrown.

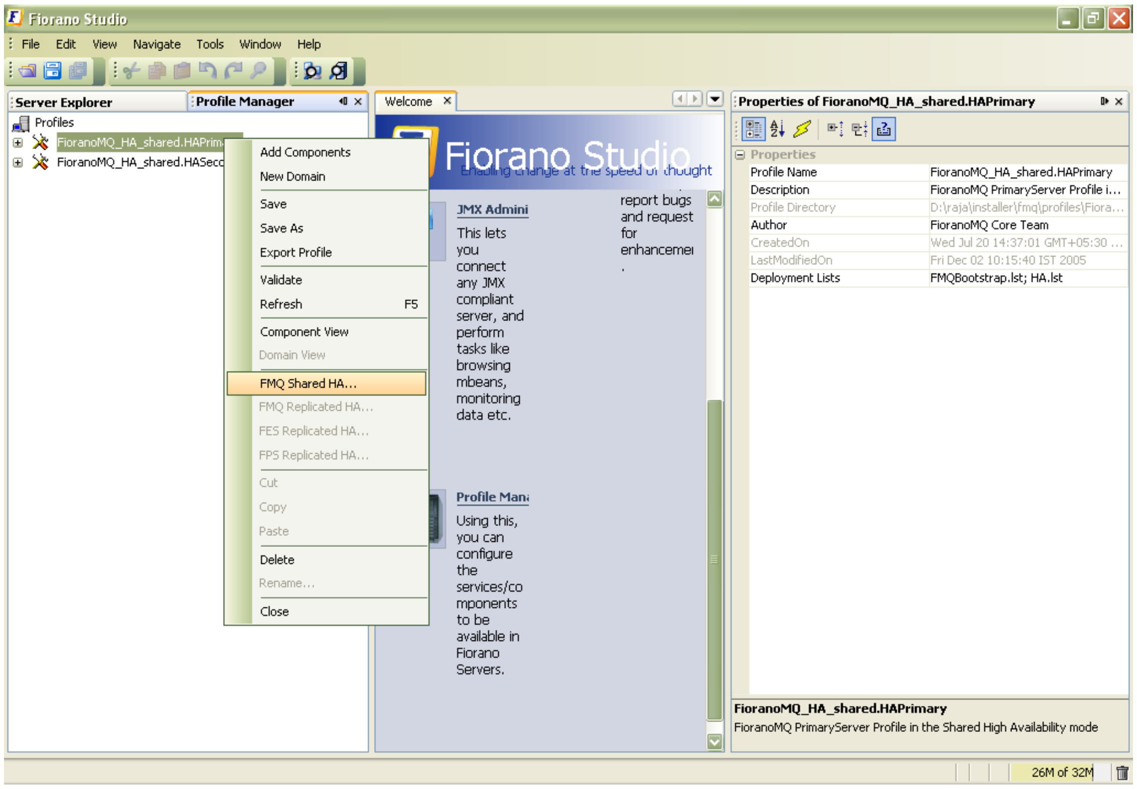
After configuration of the HA Secondary Server, the wizard should display the values, as shown in the figure below:



Property	FioranoMQ_HA_rpl.HASecondary
BackupServerIp	192.168.1.157
BackupServerPort	1856
BackupHAIPAddress	192.168.1.157
BackupHAPort	2000
Port	3000
PingInterval	30000
GatewayServerIPAddress	192.168.1.159
GatewayServerPort	7
PrimaryServerPreferred	no
ActivatePrimaryIfBothStandAlone	yes
GatewayListenerNotUpMesg	Connection refused
SocketCreationTimeout	10000

9.2.2.2 Configuring FMQ Shared HA

1. If a Shared HA needs to be used, open the **FioranoMQ_HA_shared / HAPrimary** and the **FioranoMQ_HA_shared / HASecondary** profiles in the Studio **Profile Manager**. Right-click on the profile, and click on **FMQ Shared HA** to display the **FMQ Shared HA** wizard.



This lets you connect any JMX compliant server, and perform tasks like browsing mbeans, monitoring data etc.

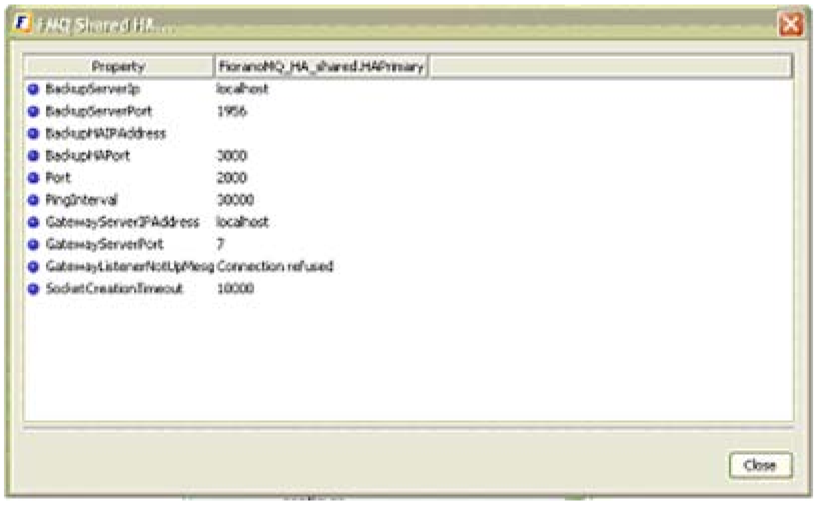
report bugs and request for enhanceme .

Using this, you can configure the services/components to be available in Fiorano Servers.

Profile Name	FioranoMQ_HA_shared.HAPrimary
Description	FioranoMQ PrimaryServer Profile i...
Profile Directory	D:\raja\installer\fmq\profiles\Fiora...
Author	FioranoMQ Core Team
CreatedOn	Wed Jul 20 14:37:01 GMT+05:30 ...
LastModifiedOn	Fri Dec 02 10:15:40 IST 2005
Deployment Lists	FMQBootstrap.lst; HA.lst

FioranoMQ_HA_shared.HAPrimary
FioranoMQ PrimaryServer Profile in the Shared High Availability mode

2. All required values can be configured using the wizard. Upon completing configuration, right-click the node, and select the **Save** option from the pop-up menu.


FMQ Shared HA...

Property	FioranoMQ_HA_shared.HAPrimary
BackupServerIp	localhost
BackupServerPort	1956
BackupHAIPAddress	
BackupHAPort	3000
Port	2000
PingInterval	30000
GatewayServerIPAddress	localhost
GatewayServerPort	7
GatewayListenerNotUpMesg	Connection refused
SocketCreationTimeout	10000

The following properties are available in the FMQ HA Replicated wizard.

BackupServerIp

This IP represents the URL where the FioranoMQ Backup Server runs. In the event of a server failure, clients connect to the Backup Server.

BackupServerPort

This port represents the backup machine server port on which the Backup FioranoMQ Server runs.

BackupHAIPAddress

This IP represents the URL of the additional FioranoMQ HA Server. In the event that this HA Server fails, all required synchronized data is available in the Backup HA Server and HA Layer that connects to Backup HA Server.

BackupHAPort

This port represents the Backup machine HA port on which another FioranoMQ Server runs.

Port

This port represents the FMQ server HA port on which the HA Server runs.

PingInterval

This interval represents the duration within which the HA Server pings the HA Backup Server to determine whether the server is running.

GatewayServerIPAddress

This value represents the IP address of the third machine that is always present in the network. This value must not represent the machine on which the FioranoMQ Servers run.

GatewayServerPort

This value represents the port where the HA server pings. This port cannot be used for another purpose.

ActiveLockAcquisitionInterval:

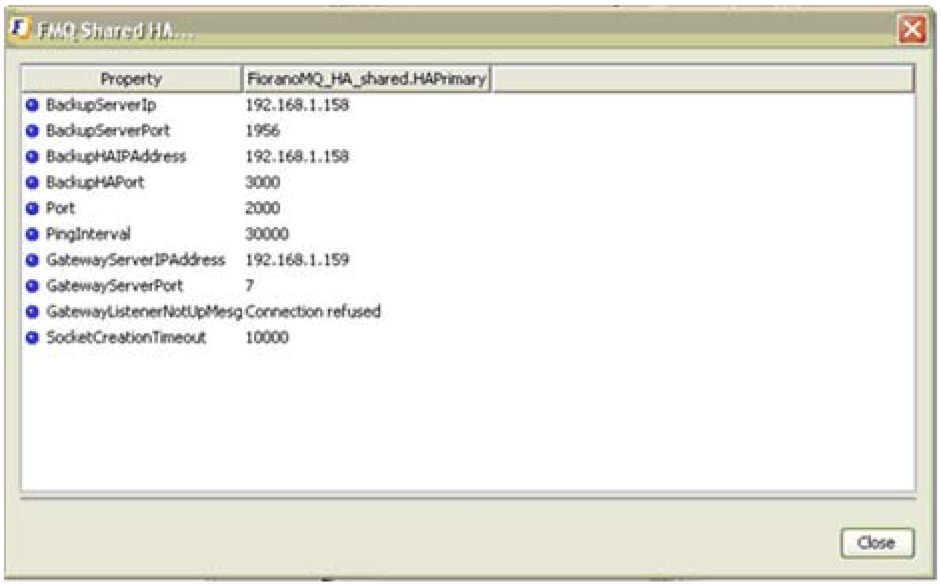
This parameter indicates the wait interval between each attempt to acquire the lock for Active Server. This value should be in multiples of half of the PingInterval value (PingInterval / 2). Otherwise, the server may try to acquire the lock on the next multiple of this value

PassiveLockAcquisitionInterval:

This parameter indicates the wait Interval for the Passive Server to acquire the lock when the link between active and Passive Server is down. This value should be greater than '2***ActiveLockReacquisitionInterval'**, otherwise an exception will be thrown. The server will not start if this value is not set properly.

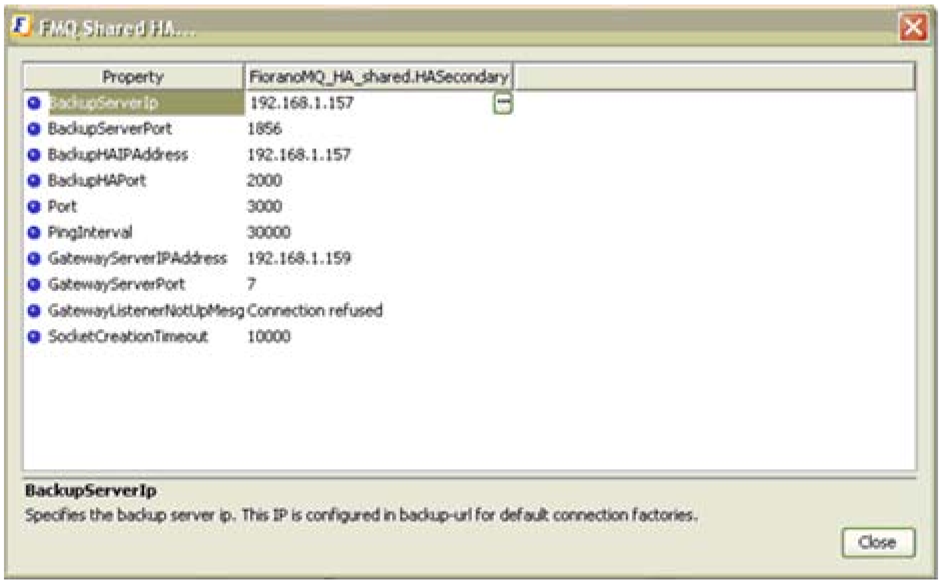
All IP addresses and ports should represent the correct values. Even without the modification of any of the values, the HA Primary Server and the HA Secondary Server are able to run on a single machine.

For example in running the HA Primary Server on a machine with IP Address: 192.169.1.157 and the HA Secondary Server on a machine with IP Address: 192.169.1.159, the Gateway Server is given as 192.169.1.159 which is present in the network. After configuration, the HAPrimary wizard looks as shown below:



Property	FioranoMQ_HA_shared.HAPrimary
BackupServerIp	192.168.1.158
BackupServerPort	1956
BackupHAIPAddress	192.168.1.158
BackupHAPort	3000
Port	2000
PingInterval	30000
GatewayServerIPAddress	192.168.1.159
GatewayServerPort	7
GatewayListenerNotUpMesg	Connection refused
SocketCreationTimeout	10000

The HA Secondary wizard displays:



Property	FioranoMQ_HA_shared.HASecondary
BackupServerIp	192.168.1.157
BackupServerPort	1856
BackupHAIPAddress	192.168.1.157
BackupHAPort	2000
Port	3000
PingInterval	30000
GatewayServerIPAddress	192.168.1.159
GatewayServerPort	7
GatewayListenerNotUpMesg	Connection refused
SocketCreationTimeout	10000

BackupServerIp
Specifies the backup server ip. This IP is configured in backup-url for default connection factories.

9.2.3 Step 3: Configuring Admin Store

To configure **Admin Store**, decide on the type of Shared Admin Object Store to be used with the FioranoMQ Server. Choose from the options given below:

- File Based (Fiorano's proprietary file format) – default option
- RDBMS Server
- XML File
- LDAP Server

Fiorano recommends the RDBMS or LDAP Server, as per instructions provided in *chapter 2*.

Naming Manager

Note*:* If the default implementation or xml-based implementation in Shared HA mode is used, the path attribute for the **Naming Manager** Server should point to the same physical location. This requires mapping a drive locally on the machine, given that both servers that are a part of the Enterprise Server are running on separate machines.

9.2.4 Step 4: Configuring Common Security Store

The *FioranoMQ Security Realm* can be categorized into two domains.

- **Principal** – user management
- **ACL –** Access Control List

The FioranoMQ Server provides a pluggable component for both the above domains. These components, namely, the Principal Manager and ACL Manager, provide user management and ACL Management services to the server respectively. Both of these components require a data Store in order to store the security information that they manage.

The options for storage media are:

- File Based (Fiorano's proprietary file format) – default option
- RDBMS Server
- XML File
- LDAP Server

When configuring a FioranoMQ Server in HA Shared mode, the Data Store used by both the servers (in an Enterprise Server) should point to the same physical location. When setting up the Enterprise Server, it is recommended that an RDBMS or an LDAP server be used as the store for security managers.

Note: Instructions for configuring *Security Realms* to use the above types of storage media can be found in the Chapter *FioranoMQ Security*.

If using default implementation or xml-based implementation in Shared HA mode, the path attribute for the Principal Manager or the ACL Manager should point to the same physical location. This requires mapping a drive locally on a machine, given that servers that are part of the Enterprise Server are running on separate machines.

9.2.5 Step 5: Configuring Database

For storing messages, the FioranoMQ Server provides the option of using either an RDBMS server or a file based database. When using an Enterprise Server, both of the servers are required to point to the same database (RDBMS or File Based).

Instructions for configuring both types of databases are given below.

RDBMS Server:

Note: Refer to *Chapter 5: Configuring Message Store* for instructions on setting up a single instance of FioranoMQ so as to use an RDBMS Server as the back-end message store.

Instructions for several commonly used databases are provided. When setting up an Enterprise Server, configure both the server instances with the same set of database parameters.

When using an RDBMS Server, ensure that the Data Store of the destination being used as RDBMS is specified, instead of the default File-based Data Store.

9.2.5.1 File Based DataStore

In the default mode, FioranoMQ creates a proprietary file-based database for storing messages. This database is created by default in the run folder of the profile being used.

For example, in the default profile, the database directory can be navigated to, using the path below:

%FMQ_DIR%\fmq\profiles\FioranoMQ\run

Within an Enterprise Server, both of the servers should have their databases pointing to the same physical directory. The database directory can be specified through the command line option –fmq.db.path when running the server. For example, assuming that a shared database is to be created in \fmq-db directory, the command line to launch both the servers would be:

```
%FMQ_DIR%\bin > fmq.bat–fmq.db.path c:\fmq-db
```

Note*:* If both servers are to run on separate machines, the database directory must be made available to both of the machines by mapping the drive locally.

9.3 Launching

When launching the Enterprise Server, the launch sequence within the two servers is not important. Either of the servers (primary or backup) can be started first. On start up, the servers establish a communication path between each other, in order to exchange information regarding the proper functioning of all servers. The servers can be launched by using the **runContainer** script (available in the fmq/bin directory of the installation package) and through supplying the %SELECTED_HA_PROFILE% as the argument.

For example, when launching the Primary HA Server in Replicated mode, the command line would be:

```
fmq.bat –profile FioranoMQ_HA_rpl/HAPrimary
```

9.4 Verifying the HA Setup

On starting a FioranoMQ Server which is part of an HA Server, the server prints the debugging information about its own state (ACTIVE, PASSIVE, WAITING). The FioranoMQ Server also prints information about the Peer Server's state when it detects a change in the peer's state.

The console displays statements such as those shown below:

Local Server switched to ACTIVE

Or

Local Server switched to PASSIVE

9.5 Shutdown the Active Server

When started, one of the servers within the Enterprise Server is in ACTIVE state while the other is in PASSIVE state. Upon shutting down the Active Server, its peer will switch to *active* state.

Changes are indicated within debug statements, similar to those shown below:

```
Created MBeanServer with ID: -584310c9:1075597caaf:-8000:ashish:1
[03/Nov/2005 15:36:52]    license      INFO       The fiorano-mq.lic license for
the product MQ ver 8x is valid and its details are Type = Eval, Days left = 42,
Locale specific = false, CPU based = false, Node locked = false
RmiConnectorServer Listening Port: 1858
```

```
Local Server switched to WAITING
Profile D:\fioranodev_installer\fmq\profiles\FioranoMQ 9_HA_rpl/HAPrimary successfully
deployed on Thu Nov 03 15:37:06 GMT+05:30 2005
Old status of remote server = DEAD
New status of remote server = PASSIVE
Product Name: {FioranoMQ 9} Version: {9.1} Build No: {4000} Build Date: {November
3 2005}
Fiorano Server accepting connections at http://164.164.129.131:1856
Server Protocol = {TCP}
Local Server switched to ACTIVE
Old status of remote server = PASSIVE
New status of remote server = DEAD
Local Server switched to STANDALONE
```

9.6 Sample

Once both the Primary and the Secondary Servers are running in HA mode, in order to verify the installation, perform the following steps:

1. Open \fmq\samples\pubsub\DurableSubscribers\Publisher.java and DurableSubscriber.java in the same directory as your preferred Java IDE.
2. Include the parameter *BackupConnectURLs* in the environment being passed to *InitialContext*. This variable should point to the Backup Server's URL. For example, if the Backup Server is running on a host called 'backup-server' on port 1856, the following tag should be added to the environment.

   ```
   env.put (“BackupConnectURLs”, “http://backup-server:1856”);
   ```

   Make this modification in both Publisher.java as well as in DurableSubscriber.java .
3. Start the Durable Subscriber and Publisher applications using the following command line:

   ```
   run-client DurableSubscriber [-topicName ......]
   ```

Note: When using a central RDBMS database as the backend data store for FioranoMQ, ensure that an RDBMS based destination is used. By default, FioranoMQ creates 'primaryRDBMSTopic' during startup (if the RDBMS database is enabled) which can be used for this purpose. Alternatively, another RDBMS based topic can be created through the Admin GUI.

4. To publish a message, type a string through the console and press enter. These messages are received by the subscriber, which prints the text on the console.
5. Now, shutdown the Primary Server using the shutdown script.
6. The Secondary Server initiates its startup sequence within the HA. ping_interval as set in Fiorano Studio.
7. If another message is published before the Backup Server completes its start up, the message is not delivered to the subscriber immediately.
8. As soon as the Secondary Server starts up, the durable subscriber's connection is re-established, and any pending messages are delivered.

9.7 Logging and Tracing

9.7.1 Logging

Just like the base FioranoMQ Server, the HA Server can log files, consoles, or use any other custom-made logger. This type of logging is controlled through the Loggers module in Fiorano Studio. The option to log all information on the console or save all logs onto a log file or an error file is provided through Console-based and File-based logging, respectively.

9.7.2 Tracing

The amount of information that is logged by the HA Server can be controlled through the trace variable for HA. Upon startup, the server initializes the value of various trace components. Integers between 0-6 are valid and are considered acceptable values. Higher values of the trace variable for HA in this file results in more information logged.

```
HA=0  /** No Log                              **/

HA=1  /** Default Log                           **/

HA=2  /** Displays State Info of HA Server        **/

HA=4  /** Displays intra server communication info **/

HA=6  /** Maximum HA log                          **/
```

9.8 Limitations of HA

- Client level transactions do not span across servers in the Enterprise Server when running on the Shared mode. Transacted sessions involving Receivers are rolled back in the event that the Primary Server crashes. The messages delivered in that transaction are redelivered to the Receivers once connected to the Backup Server.
- Distributed transactions that are in execution during the transition phase become *in doubt transactions*. These transactions are rolled back and can be recovered after the client connects to the Secondary Server.
- JMS Topic Requestor cannot receive its intended reply if a failover occurs after a request is sent. This occurs because the JMS Topic Requestor creates a non-durable subscriber, which can miss a message during failover. However, if a topic requestor creates a durable subscriber to listen to replies, then it functions successfully even during a failover.
- If both HA Servers (primary as well as backup) go down, the requestor receives a duplicate reply (with redelivered Flag = true) for the request made immediately after a failover.

- Since each server has its own configuration, which includes independent Dispatcher cluster information, configuration of the dispatcher in HA is allowed only when all the member server information is provided in offline mode for each Dispatcher-enabled server in the HA pair. Since the information of member servers added to the cluster in online mode will be persisted only in the Active Server and will not be available to the Passive Server, no new nodes should be added to the cluster after the servers are started. Please note that the configuration changes are done in the Active Server and will not be replicated to the Stand-by Server, even in normal Replicated HA mode servers.

Chapter 10: Dispatcher

The FioranoMQ Dispatcher is the solution to the problem of load balancing the incoming client connections between a group of servers. The Dispatcher server is connected to multiple servers belonging to a “cluster”. The dispatcher services this “cluster”.

FioranoMQ Dispatcher maintains a persistent connection with each FioranoMQ server in a cluster. This persistent connection is used to pass information from the server to the dispatcher, enabling the dispatcher to maintain real-time **in-memory statistics** about the precise load in terms of the number of connections on each server. The dispatcher uses this information to determine the least loaded server in the cluster and routes the new incoming client requests to it.

The dispatcher is a normal FioranoMQ Server with an additional dispatcher component. Once the **dispatcher** functionality is turned **ON** in the server, it automatically routes connection requests to the least loaded server.. That is the sole difference between a plain vanilla server and a dispatcher enabled server. An advantage of using Fiorano Dispatcher is that no changes are required in the client application to use the Dispatcher since the client application is transparent to internal re-routing taking place inside the dispatcher.

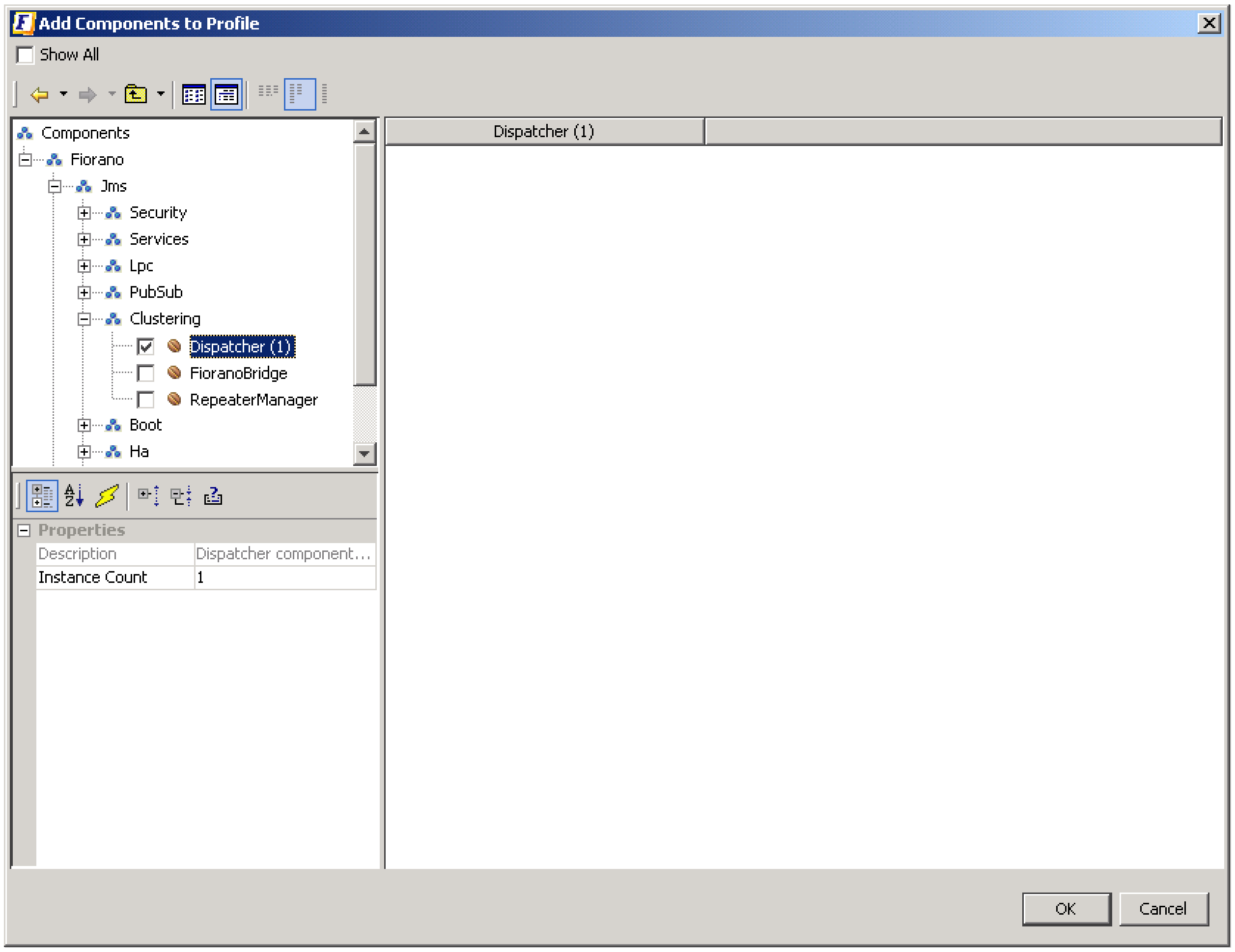
However, as mentioned before, a dispatcher server is perfectly capable of handling connection client requests as well as any MQ server. Even if none of the servers in the dispatcher's cluster are alive, the client's connection request would still be served by the dispatcher server itself.

To make use of the dispatcher functionality, the user can either use the preconfigured Dispatcher enabled FioranoMQ profile named FioranoMQ_Dispatcher or enable Dispatcher in a profile by following the steps given in the next section.

For information about configuring profiles through a text based file, see FioranoMQ Getting Started.

10.1 How to Configure Dispatcher

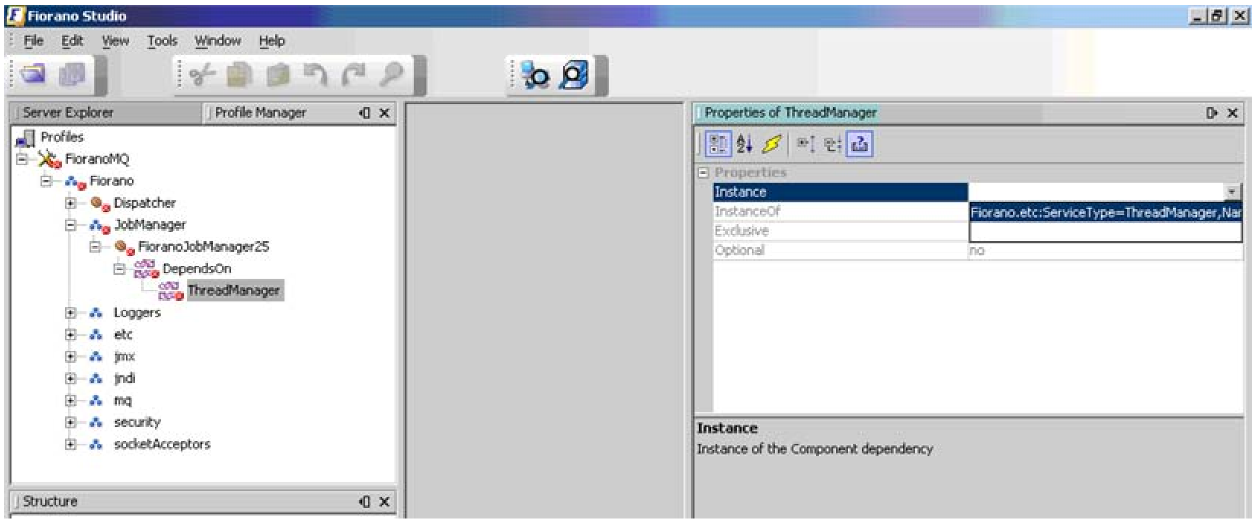
1. Launch **admin studio** and open the profile to be configured in **offline** mode.
2. Navigate to **FioranoMQ > Fiorano**. Right-click and select **Add Components**. A new window opens.
3. Navigate to **Components > Fiorano > JMS > Clustering**, and select **Dispatcher**. Click the **OK** button.



Properties

Description	Dispatcher component...
Instance Count	1

4. In the **studio** window, navigate to **FioranoMQ > Fiorano > JobManager> FioranoJobManager25> DependsOn > ThreadManager**. In the Properties pane, navigate to the '**Instance**' property and select the **ThreadManager**.

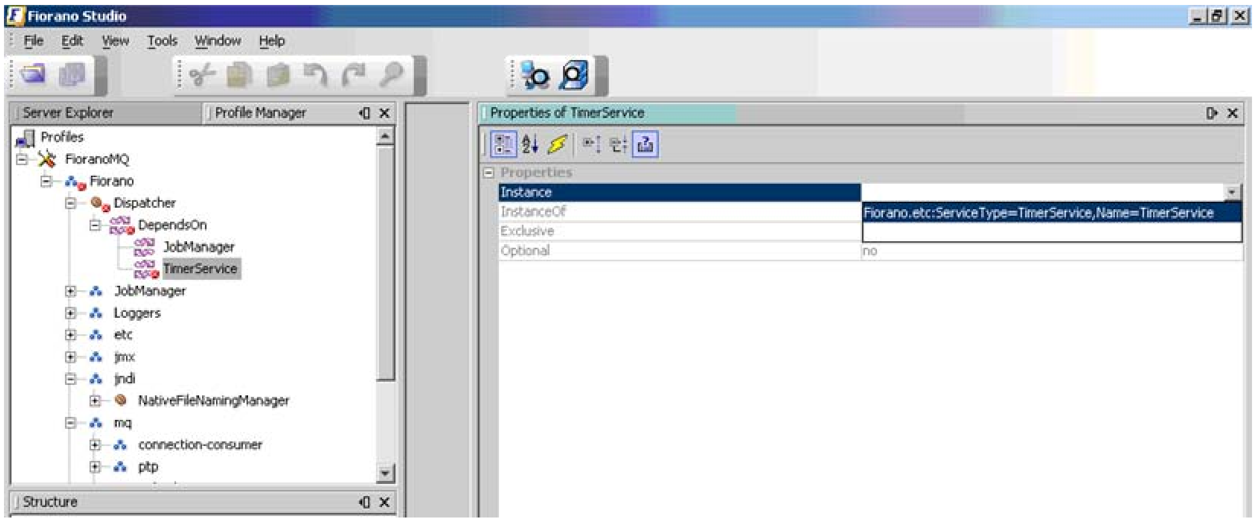


Properties

Instance	
InstanceOf	Fiorano.etc:ServiceType=ThreadManager,Nar
Exclusive	
Optional	no

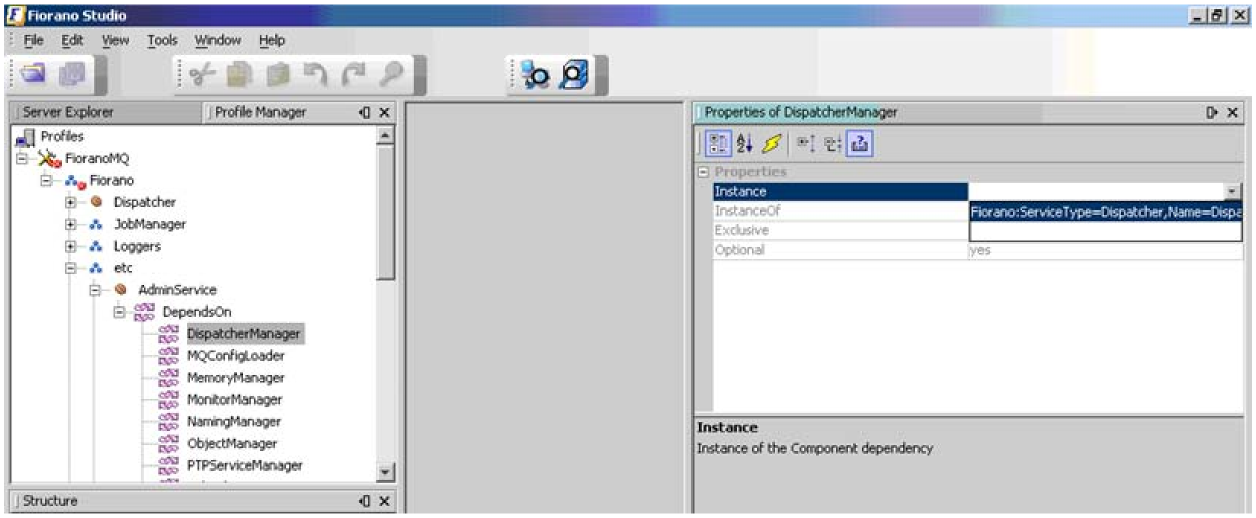
Instance
Instance of the Component dependency

5. Navigate to **FioranoMQ>Fiorano>Dispatcher>DependsOn>TimerService.** In the properties pane, navigate to the **Instance** property and select the timer service.



Instance	
InstanceOf	Fiorano.etc:ServiceType=TimerService,Name=TimerService
Exclusive	
Optional	no

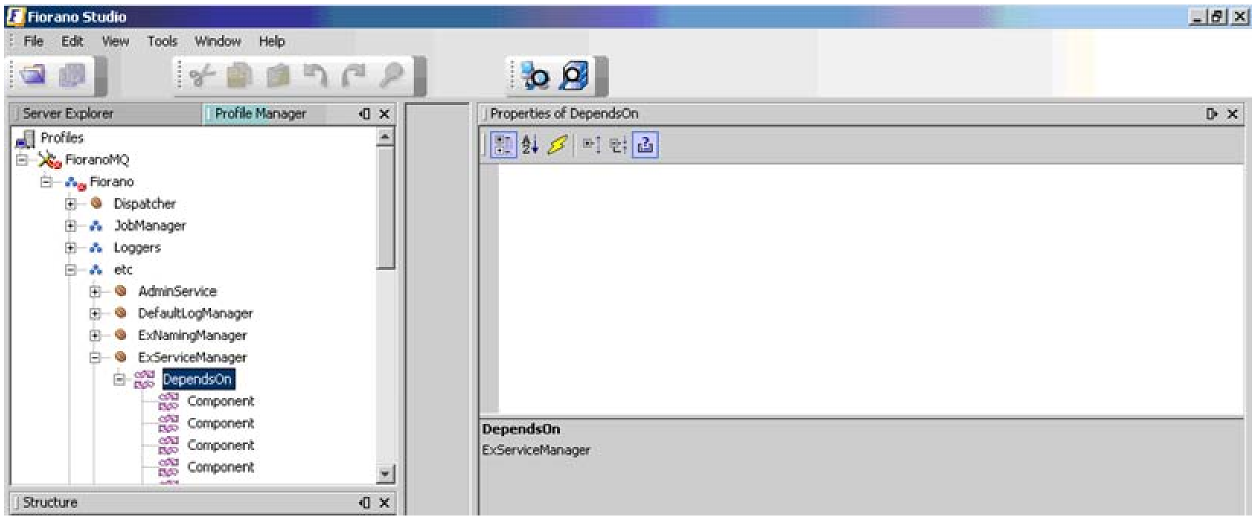
6. Navigate to **FioranoMQ > Fiorano > etc> AdminService > DispatcherManager**. In the properties pane, navigate to the **Instance** property and select the **DispatcherManager**.



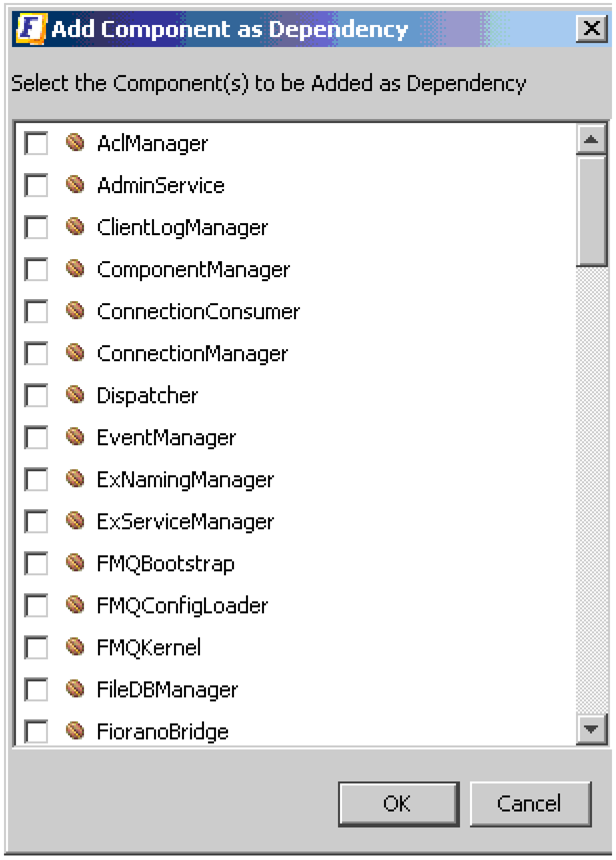
Instance	
InstanceOf	Fiorano:ServiceType=Dispatcher,Name=Dispa
Exclusive	
Optional	yes

Instance
Instance of the Component dependency

7. Navigate to **FioranoMQ > Fiorano > etc> ExServiceMaanger > DependsOn** and right-click on **DependsOn**. From the drop-down list that appears, select **Add Component**. A new window appears. From this window select **Dispatcher** and press **OK**.



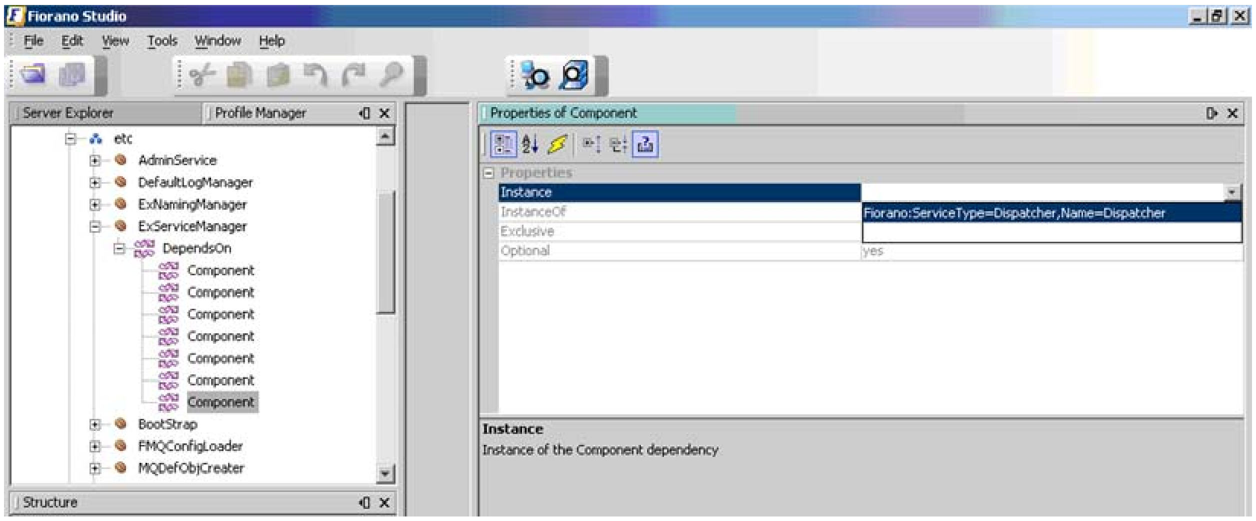
DependsOn
ExServiceManager



Select the Component(s) to be Added as Dependency

- AclManager
- AdminService
- ClientLogManager
- ComponentManager
- ConnectionConsumer
- ConnectionManager
- Dispatcher
- EventManager
- ExNamingManager
- ExServiceManager
- FMQBootstrap
- FMQConfigLoader
- FMQKernel
- FileDBManager
- FioranoBridge

8. Navigate to the added **Dispatcher** component in the **ExServiceManager**. In the properties pane, navigate to the **Instance** property, and select **Dispatcher**.



Properties

Instance	
InstanceOf	Fiorano:ServiceType=Dispatcher,Name=Dispatcher
Exclusive	
Optional	yes

Instance
Instance of the Component dependency

10.2 Adding Servers to Dispatcher Cluster

To add the dispatcher component to a profile follow the steps below:

1. Navigate to **FioranoMQ > Fiorano> Dispatcher** and right-click on it. From the drop-down, select **Add**. Select either the **preferred serve**r tab or the **server** tab. Selecting the **preferred server** tab creates an entry under **Dispatcher** for the preferred server in the dispatcher cluster. Selecting the **server** tab creates an entry under **Dispatcher** for another server in the dispatcher cluster. Multiple servers may be configured by adding them to the dispatcher cluster.
2. Navigate to **FioranoMQ > Fiorano> Dispatcher > Server**, and fill in the appropriate details for that particular server. Follow the same procedure to configure all the servers (including the preferred server) in the dispatcher cluster.

The specific details of a particular server are as follows:

Name	**Description**
Name	Represents the name of the Server
BackupUrl	Specifies the backup url in case the Server given in Url parameter is down.
LoginName	Represents the login name used by the dispatcher to connect to the Member MQ server. The login should have admin privileges.
Password	Represents the password used by the dispatcher to connect to the Member MQ server.
AdminConnectionFactory	Specifies the Admin Connection Factory used by the dispatcher to connect to the Member MQ server.
MaximumConnection	Specifies the weight associated with a member server of a cluster. A member server with MaxClientConnections set to 2 allows twice the number of connections that can be created by member server with weight of ‘1’.

Name	Description
Url	Specifies the URL of the server in the cluster (Format: http://hostname:port).

10.3 Configuring Client Applications to Use Dispatcher

The advantage of using the Fiorano Dispatcher configuration is that no changes are required on the Client end to connect through the Dispatcher server. The Dispatcher server internally routes requests to the least loaded server (server load is calculated internally by the dispatcher based on the maximum connections allowed on a particular server, and the number of active connections), rendering the client application wholly unaware of the final MQ server that it will be connected to. The server to which the createConnection call of an application has been routed may be determined using the APIs below. The APIs below may also be used to get connectURLonce the connection has been established by the User.

ConnectionMetaData connectionMetaData = connection.getMetaData();

ConnectionFactoryMetaData connectionFactoryMetaData = ((MQConnectionMetaData) connectionMetaData).getConnectionFactoryMetaData();

Now connectionFactoryMetaData.getConnectURL() will give the ConnectURL of the server. The User has to import the following Fiorano Specific classes for calling these apis fiorano.jms.md.ConnectionFactoryMetaData, fiorano.jms.md.MQConnectionMetaData.

User can refer to the $FMQ_INSTALLER/fmq/samples/PTP(or)PubSub/Dispatcher samples foradditional information.

This API returns the string equivalent of the URL used to create the connection. At times, a given client application might want to connect to a particular server in a cluster. This can be done by setting the variable LookupPreferredServer to 'true' in the environment settings:

env.put ("LookupPreferredServer", "true");

Other statements remain the same as specified above.

10.4 Using Preferred-Server Configuration

The preferred-server is typically used by client applications that have previously created durable subscriptions on a particular FioranoMQ server within a known server cluster, and wish to reconnect to the same server to retrieve messages. The preferred server may be set using Fiorano Studio.

The FioranoMQ cluster may be set using the Fiorano Admin Studio. Log onto the FioranoMQ server (with Dispatcher enabled).A Dispatcher node in the Explorer tree is displayed which may be used to manage the FioranoMQ cluster. Servers from the cluster managed by this dispatcher may be added or removed. The status of existing servers in the cluster may be viewed. Managing the Dispatcher includes:

- Adding and removing servers connected to the cluster.
- Setting the Preferred Server in the Cluster managed by the Dispatcher

- o Setting the maximum number of client connections (as a weightage) to a Server.

Note: For more information on how to perform the above tasks using Fiorano Studio, refer to the section 23.8 Working with Dispatcher.

In addition to the above, FioranoMQ provides comprehensive APIs to control and manage the dispatcher and the cluster associated with it.

Note: Detailed information on these APIs is available in the JavaDocs provided with the FioranoMQ installation package.

Chapter 11: Repeater

11.1 Launching Repeater in Stand-Alone Mode

The Repeater can be launched in Stand-Alone Mode as described below:

Start the FioranoMQ Console.

Navigate to the %FIORANO_HOME% \fmq\bin directory.

A ready-to-use batch file to launch the Repeater is now available. Instructions for using it are given below:

To run StandaloneRepeater on windows systems: : [Platform? Or system?]

fmq.bat -profile StandAloneRepeater

To run StandaloneRepeater on non-windows systems: [Platform? Or system?]

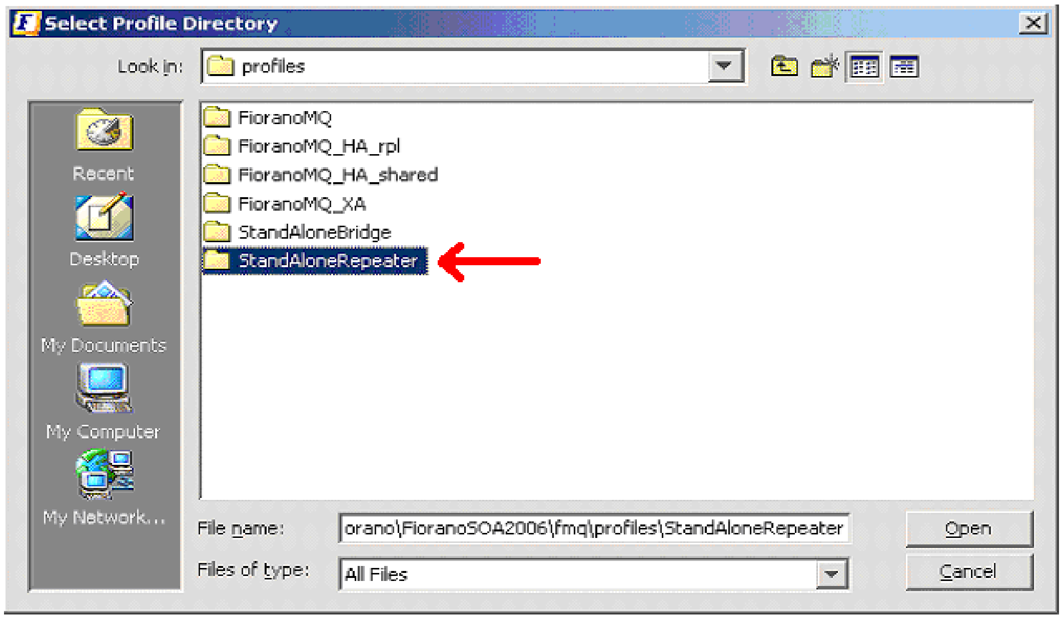
fmq.sh -profile StandAloneRepeater

11.2 Configuring Repeater in the Off-line Mode

In the offline mode, the administrator is able to add links to the repeater and configure the source and target servers for message replication. Cluster administrators are provided with a template configuration file (**Configs.xml** located in %FIORANO_HOME%\fmq\profiles\StandAloneRepeater\conf directory of the FioranoMQ installation package) to simplify the Repeater configuration. Two links are added to the Repeater configuration linking the source and the target servers bi-directionally. The tool displays the Repeater with these default links. These links can be configured as described below:

For information about configuring profiles through a text based file, see FioranoMQ Getting Started.

1. Launch Fiorano Studio using Studio.bat (or Studio.sh) located in %FIORANO_HOME%\Studio\bin
   On windows, this may be launched selecting **Start** > **Programs** > **Fiorano** > **FioranoMQ** > **Fiorano Studio** for offline configuration of the FioranoMQ server.
2. Select **Tools** > **Configure Profile** from the menu bar, and select the %FIORANO_HOME%\fmq\profiles\StandAloneRepeater directory from the **Select Profile Directory** dialog box. Click on the **Open** button.

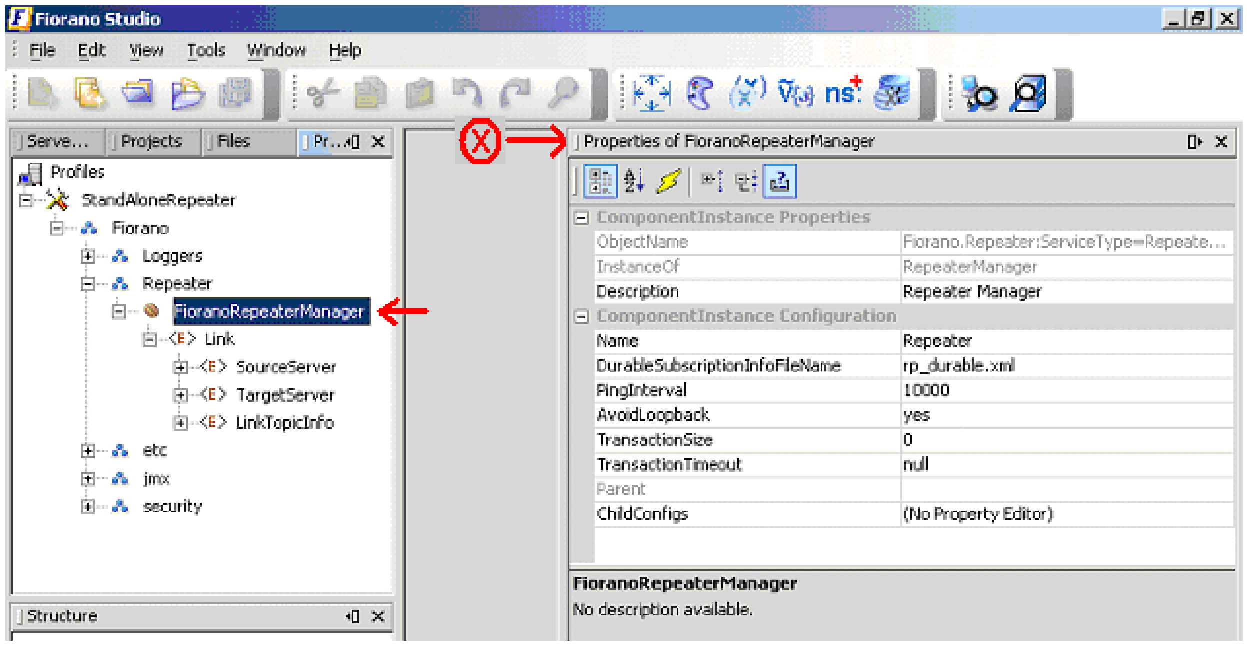


FioranoMQ
FioranoMQ_HA_rpl
FioranoMQ_HA_shared
FioranoMQ_XA
StandAloneBridge
StandAloneRepeater

File name: orano\FioranoSOA2006\fmq\profiles\StandAloneRepeater

Files of type: All Files

3. Navigate to **StandAloneRepeater > Repeater >FioranoRepeaterManager** node in the Server Explorer. The properties of the repeater are displayed in the **Properties Pane** (marked 'X' in the figure). Configure them as required.



Profiles
- StandAloneRepeater
  - Fiorano
    - Loggers
    - Repeater
      - FioranoRepeaterManager
        - <E> Link
          - <E> SourceServer
          - <E> TargetServer
          - <E> LinkTopicInfo
    - etc
    - jmx
    - security

ComponentInstance Properties	
ObjectName	Fiorano.Repeater:ServiceType=Repeate...
InstanceOf	RepeaterManager
Description	Repeater Manager
ComponentInstance Configuration	
Name	Repeater
DurableSubscriptionInfoFileName	rp_durable.xml
PingInterval	10000
AvoidLoopback	yes
TransactionSize	0
TransactionTimeout	null
Parent	
ChildConfigs	(No Property Editor)

FioranoRepeaterManager
No description available.

11.2.1 Editing a Link

You can edit the properties related to this default link before creating and managing additional links in the online mode. The Link element within the Repeater Manager MBean contains the following elements:

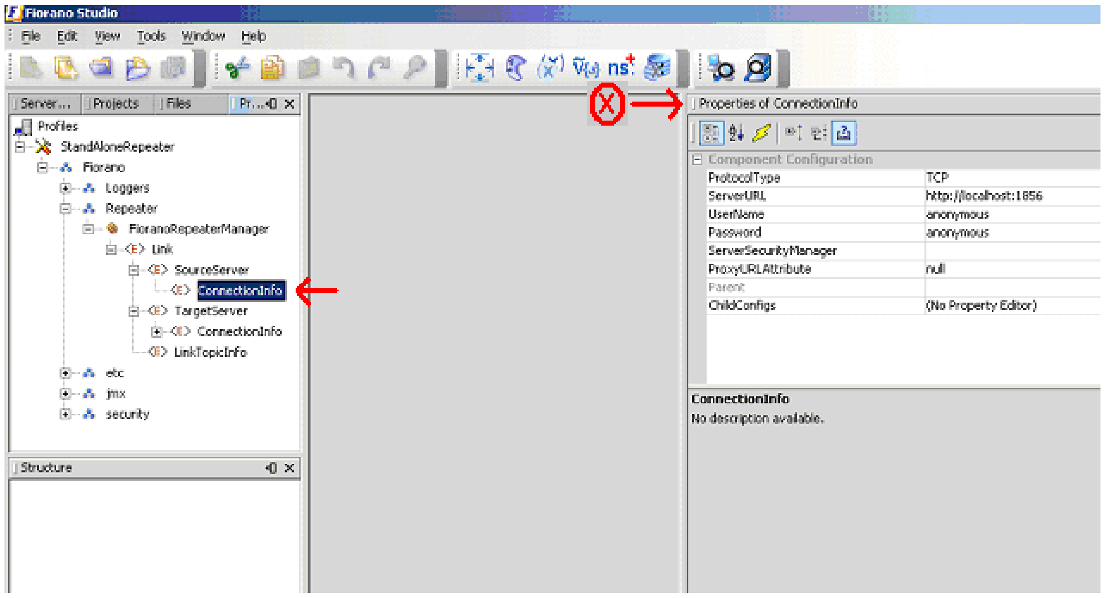
Status - Specifies whether the link is running or not.

SourceServer - Specifies the server on which subscriptions are created. The Source Server contains the ConnectionInfo.

TargetServer - Specifies the Server on which publishers are created. The TargetServer contains the ConnectionInfo.

To edit:

1. Navigate to the desired Link and expand its components.
2. Selecting a link causes its properties to be displayed in the Properties Pane (marked 'X' in the figure), where the corresponding Properties may be edited.



Component Configuration

ProtocolType	TCP
ServerURL	http://localhost:1856
UserName	anonymous
Password	anonymous
ServerSecurityManager	
ProxyURLAttribute	null
Parent	
ChildConfigs	(No Property Editor)

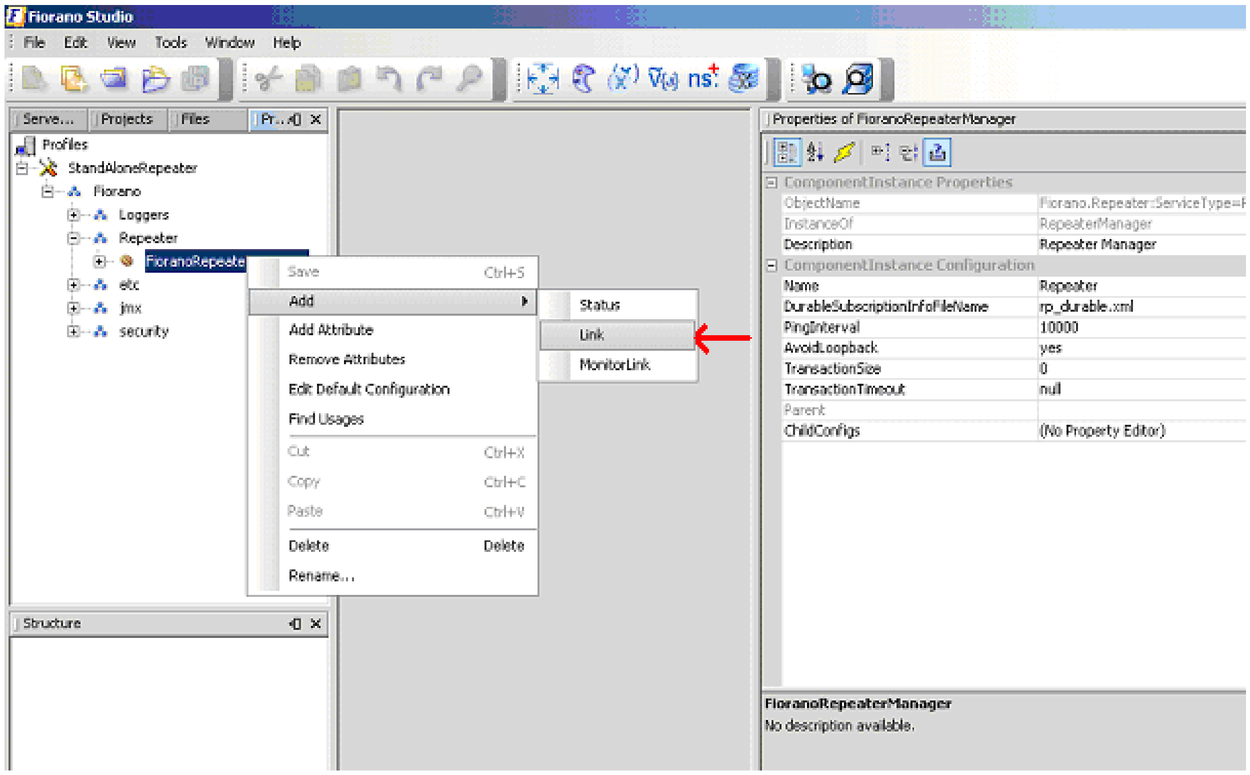
ConnectionInfo
No description available.

3. **Save** the configuration for changes to take effect.

11.2.2 Adding a Link

The repeater replicates messages in the link specified between the source and the target server. A repeater can have a number of links configured. By default, the server sets up only a single link to the repeater. A new link may be added to the Repeater when it configured in offline mode, as shown in the steps below:

1. Navigate to StandAloneRepeater > Repeater >FioranoRepeaterManager node in the Server Explorer. Right-click the FioranoRepeaterManager node and select Add > Link option from the short-cut menu.



Properties of FioranoRepeaterManager

ComponentInstance Properties

ObjectName	Fiorano.Repeater:ServiceType=F
InstanceOf	RepeaterManager
Description	Repeater Manager

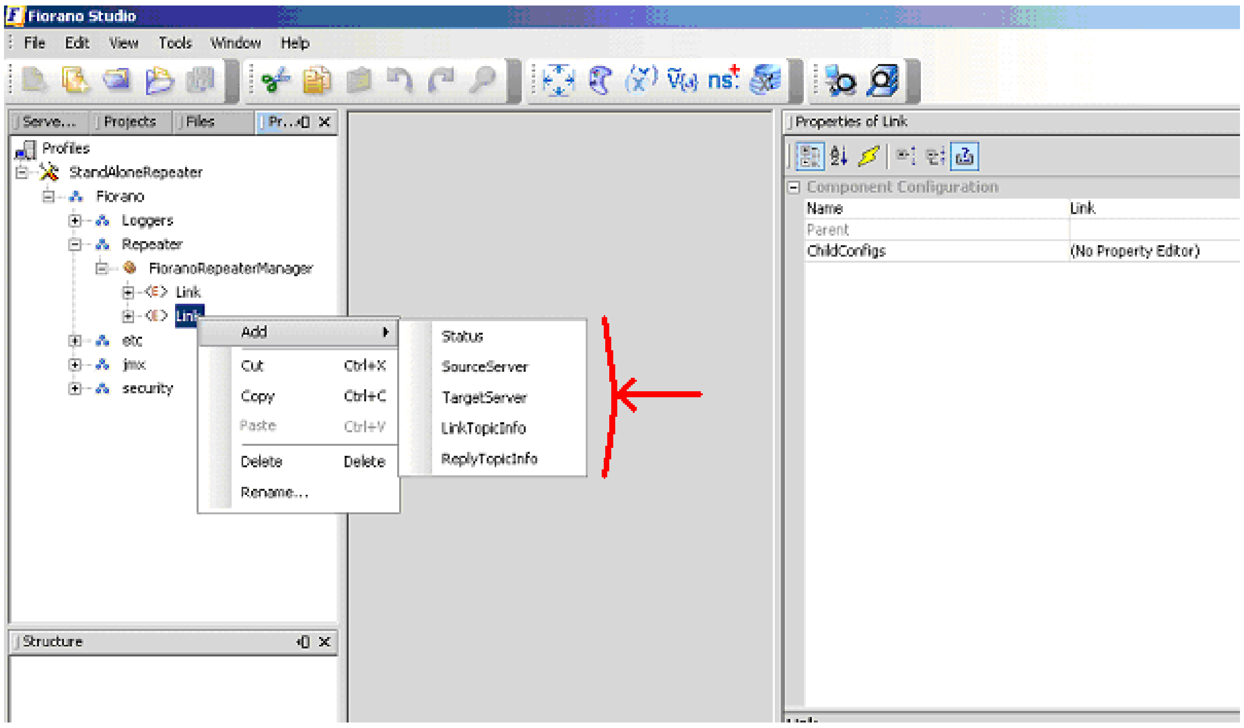
ComponentInstance Configuration

Name	Repeater
DurableSubscriptionInfoFileName	rp_durable.xml
PingInterval	10000
AvoidLoopback	yes
TransactionSize	0
TransactionTimeout	null
Parent	
ChildConfigs	(No Property Editor)

FioranoRepeaterManager
No description available.

2. A new Link can be added to the **FioranoRepeaterManager**.
3. Other parameters, such as **Status**, **SourceServer**, **TargetServer**, **LinkTopicInfo**, and **ReplyTopicInfo** may be added to this Link.
4. Provide unique names for each link, source & target servers.
5. Provide the Username/Password in the connectionInfo of the source and target server, the specified user need not have admin privilege but must have access to the topic (i.e., +ve perm in topics ACL).

Separate links are to be provided for each topic which has different ACLs.



Name	Link
Parent	
ChildConfigs	(No Property Editor)

6. **Save** the configuration to render it effective and embed the changes into Configs.xml file.

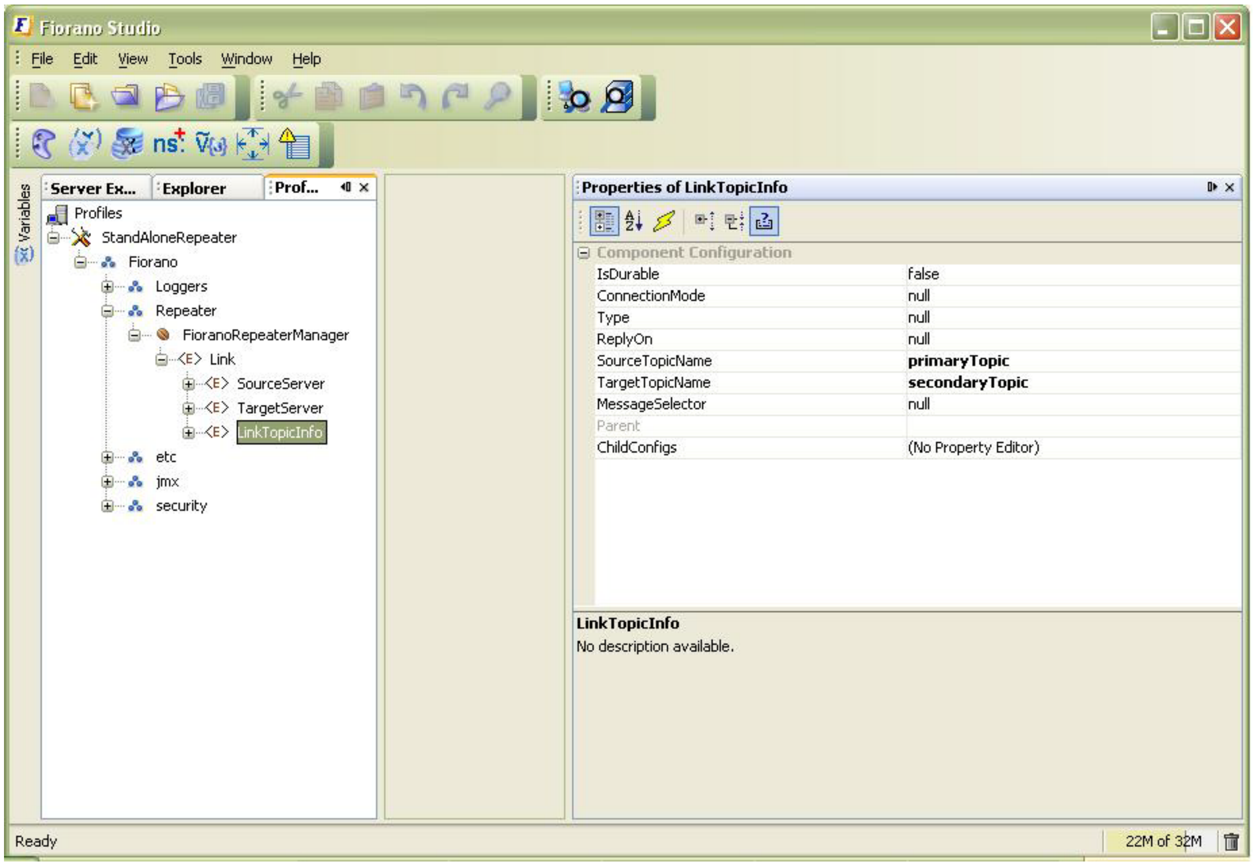
11.2.3 Deleting a Link

Unwanted Links may be deleted by executing the steps given below:

1. Navigate to the Link to be deleted and right-click on it.
2. Select **Delete** from the shortcut menu. A **Confirm Object Deletion** dialog box is displayed. Click on the **Yes** button.

11.2.4 Adding a Topic Propagation Link

To add a LinkTopicInfo, right-click on the **Link** node and select **LinkTopicInfo** from the shortcut menu. A new LinkTopicInfo is added to the selected link. The properties for the LinkTopicInfo are displayed in the **Properties Pane**.



Properties of LinkTopicInfo

Component Configuration

IsDurable	false
ConnectionMode	null
Type	null
ReplyOn	null
SourceTopicName	**primaryTopic**
TargetTopicName	**secondaryTopic**
MessageSelector	null
Parent	
ChildConfigs	(No Property Editor)

LinkTopicInfo
No description available.

A description of the LinkTopicInfo properties is given below:

- **IsDurable*:*** Specifies whether the link between the source and target is durable or not. A durable link can be used to ensure that no messages are lost across the repeater in case of network failure. The possible values for this variable are **yes** and **no**

- **ConnectionMode*:*** This parameter specifies whether same JMS connection should be used for replicating the data across JMS servers or whether a separate connection for each link is needed. The two possible values for this are 'shared'(default) or 'exclusive'

- **Type*:*** Specifies whether the link should be permanently connected to the target server or only replicate if a subscriber exists. The two possible and valid entries for this property are SUBSCRIBEREXISTS and ALWAYS.

- **ReplyOn*:*** Specifies the topic name on which the repeater listens for the replies which it receives for the requests it forwards.

- **SourceTopic Name*:*** Specifies the name of the topic on which subscriptions are made on the source server of the link. This name supports the wild character, '*', which, if specified, enables the repeater to create subscriptions on all the topics that match the source Topic. For the source topic name 'ABC*', subscriptions are made on topics ABC1, ABC12, ABCDEF and so on.

- **Target Topic Name*:*** The name of the topic on which messages received for the above subscription are forwarded onto the target server of the link.

- **Message Selector*:*** Specifies the selector that is set on a link between servers so that only messages that are required/necessary are exchanged between them.

11.2.5 Deleting or Editing a Topic Propagation Link

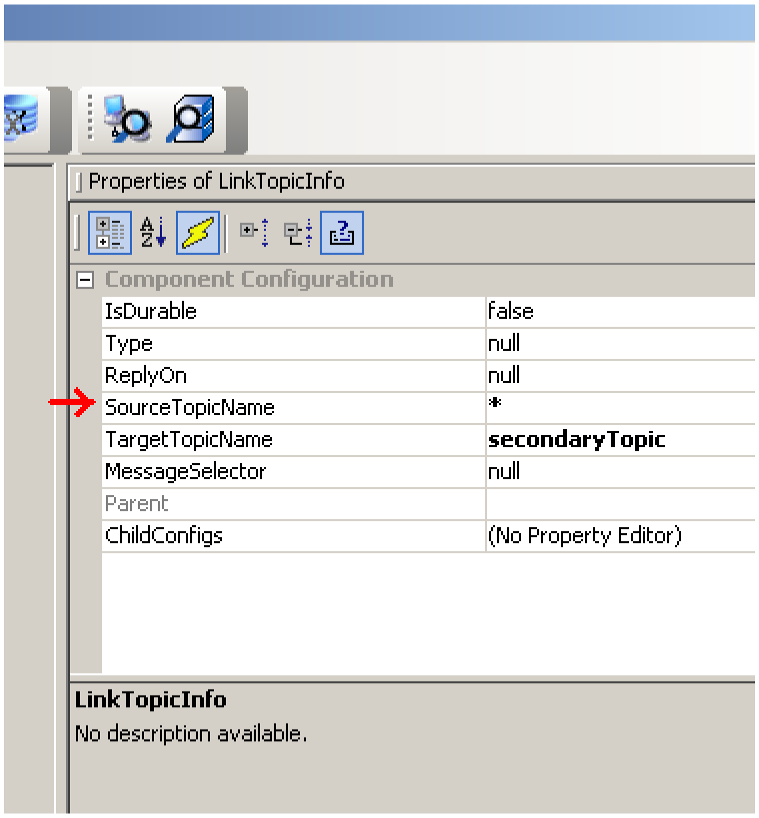
Editing a Topic Propagation Link can be done by changing values in the properties parameters, as shown in the figure above.

To delete a Topic Propagation Link, right-click on **LinkTopicInfo** and select **Delete** from the shortcut menu. A **Confirm Object Deletion** dialog box is displayed. Click on the **Yes** button.

11.2.6 Hierarchical Topics

11.2.6.1 Wild Character Support

FioranoMQ provides support for wild-card characters such as '*' within the repeater configuration so that separate links need not be added for each topic in turn. A user can specify wild-card characters in the source topic. All topics starting with the string mentioned in the source topic may be repeated.



Component Configuration

IsDurable	false
Type	null
ReplyOn	null
SourceTopicName	*
TargetTopicName	**secondaryTopic**
MessageSelector	null
Parent	
ChildConfigs	(No Property Editor)

LinkTopicInfo
No description available.

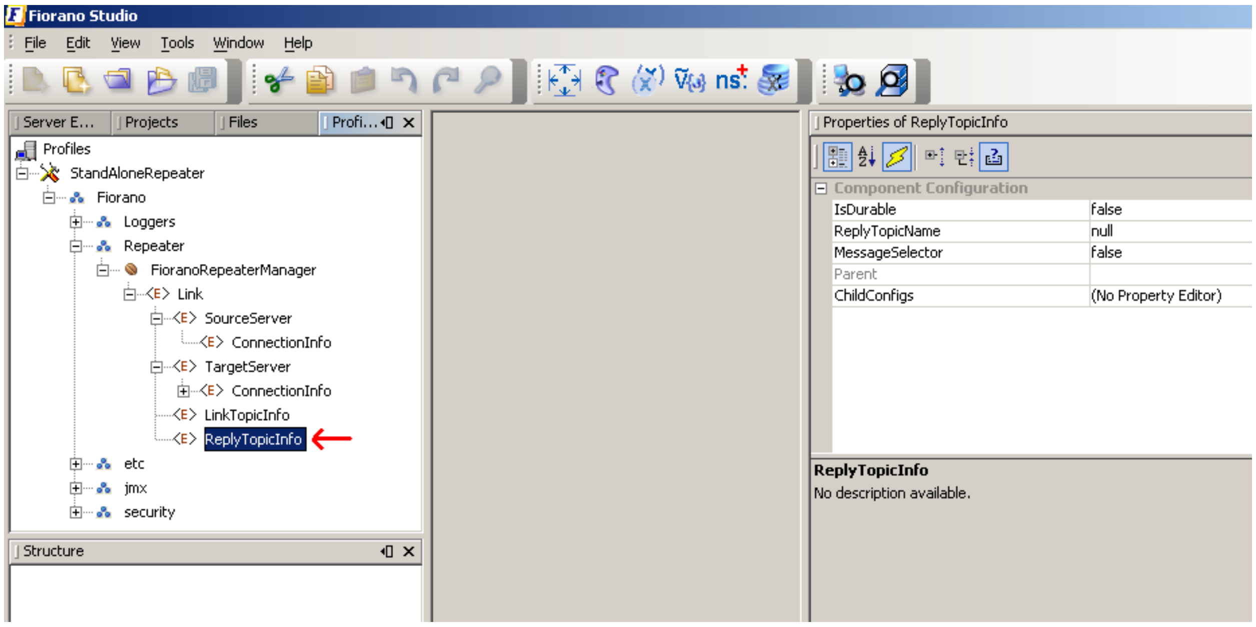
11.2.6.2 Replicate topics with a pattern

The repeater may be configured to replicate messages that match a particular pattern. The pattern can be specified in the source topic name located in the Properties Pane. For example, if the Source Topic Name is “ABC*”, the topics which match this pattern (all the topics starting with the string “ABC” on the source server) are repeated across two servers. Hence, all subscribers subscribing on ABC, ABC1, ABCZ and so on are able to receive messages published on source topics ABC, ABC1 and ABCZ respectively, via the FioranoMQ repeater. Dynamically created topics that match the pattern ‘ABC*’ are replicated. For example, if ‘ABC2’ is created after the repeater has started, a replication link for ‘ABC2’ (topic on source server) to ‘ABC2’ (topic on target server) is created dynamically. If a topic name that does not match the pattern (such as ‘ABD1’), is created, the replication link (‘ABC*’) is not be added.

11.2.7 Configuring Request/Reply through Repeater

11.2.7.1 Adding a Reply Topic Link

To add **ReplyTopicInfo**, right-click the **Link** node and select **ReplyTopicInfo** from the shortcut menu. A new ReplyTopicInfo is added to the selected link. The properties for the ReplyTopicInfo are displayed in the **Properties Pane**.



IsDurable	false
ReplyTopicName	null
MessageSelector	false
Parent	
ChildConfigs	(No Property Editor)

ReplyTopicInfo
No description available.

A description of the ReplyTopicInfo properties is given below:

- ***IsDurable:*** Specifies whether the link between the source and target is durable. A durable link can be used to ensure that no messages are lost across the repeater in the event of a network failure. The possible values for this variable are **yes** and **no**.

- **ReplyTopicName:** Specifies the name of the ReplyTopic.

- **Message Selector*:*** Specifies the selector that is set on a link between servers so that only the required/necessary messages are exchanged between the servers.11.2.7.2 Deleting or Editing a Reply Topic Link

Editing a Reply Topic Link can be done by changing the values in the properties parameters, as shown in the figure above.

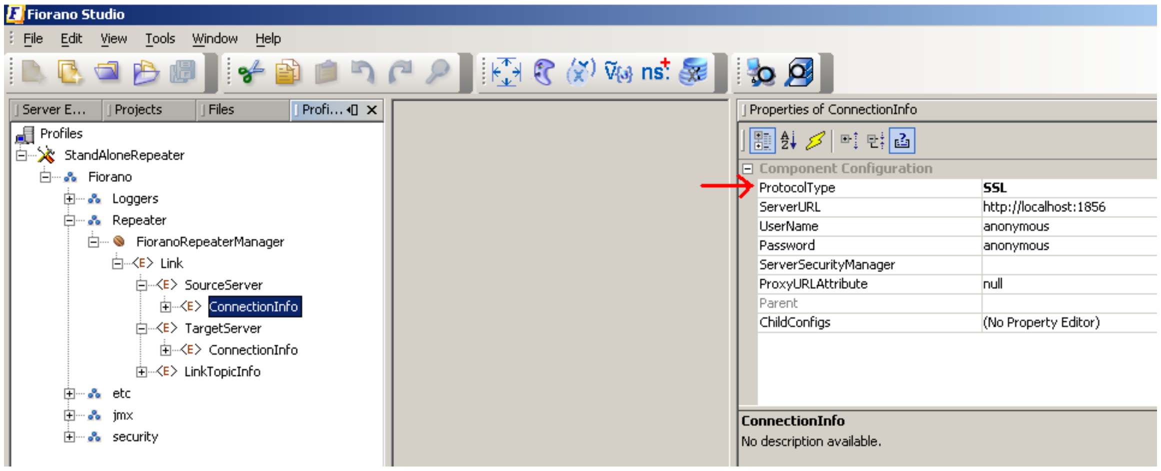
To delete a Reply Topic Link, right-click on **ReplyTopicInfo** and select **Delete** from the shortcut menu. The **Confirm Object Deletion** dialog box is displayed. Click on the **Yes** button.

11.2.8 Running a Repeater on secure protocol

FioranoMQ supports servers that run on a secure protocol. The Repeater may be run on a secure protocol so as to connect to a FioranoMQ server running on a secure protocol. The parameters listed below need to be configured in order to run the FioranoMQ Repeater on a protocol with security:

The Protocol Type of Source/Target servers must be the same as the protocol on which the Server is running. Possible values are:

- TCP with no security
- HTTP with no security
- SUN_SSL TCP with JSSE security
- HTTPS_SUN HTTP with JSSE security



Component Configuration

ProtocolType	SSL
ServerURL	http://localhost:1856
UserName	anonymous
Password	anonymous
ServerSecurityManager	
ProxyURLAttribute	null
Parent	
ChildConfigs	(No Property Editor)

ConnectionInfo
No description available.

ServerSecurityManager must be set to the name of the class that is used in the process of authenticating the client with the server. By default, FioranoMQ runtime provides one implementation each for JSSE enabled SSLs.

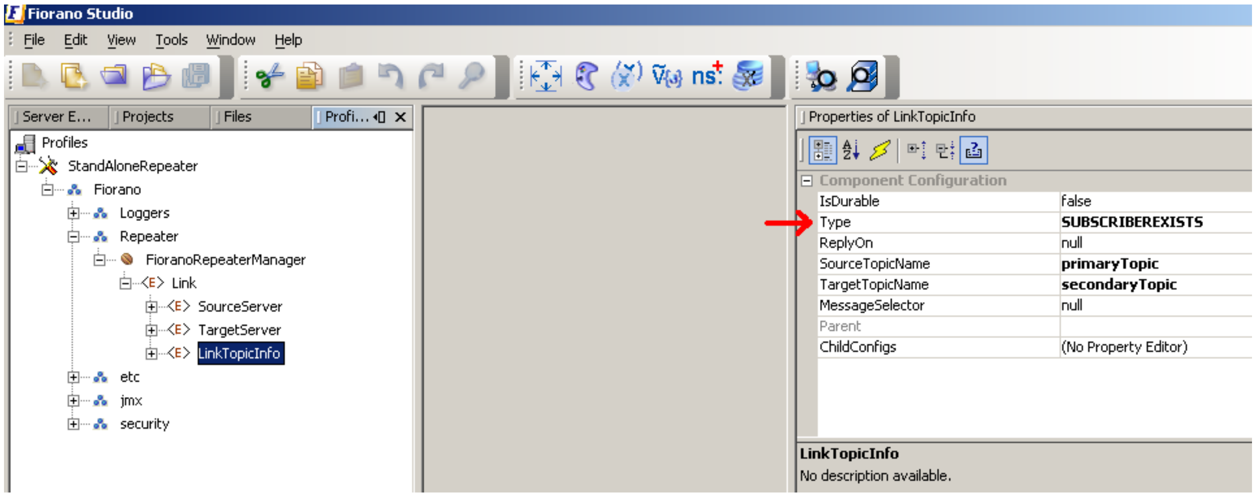
fiorano.jms.runtime.sm.JSSESecurityManager for JSSE.

Change these properties in the repeater and save all the changes.

When the security enabled repeater runs, it connects to the server running on a secure protocol so as to replicate messages.

11.2.9 Configuring Replication on Demand

To configure replication on demand, the Type property available with **LinkTopicInfo** applicable to the target server should be set as SUBSCRIBEREXISTS. If this is not done, it remains passive.



IsDurable	false
Type	SUBSCRIBEREXISTS
ReplyOn	null
SourceTopicName	primaryTopic
TargetTopicName	secondaryTopic
MessageSelector	null
Parent	
ChildConfigs	(No Property Editor)

LinkTopicInfo
No description available.

11.2.10 Configuring Monitoring Support

FioranoMQ comes with two different types of Connector mechanisms that can be used to monitor and administer FioranoMQ or its tools such as the dispatcher/bridge/repeater etc. The Connectors are the:

- RMI Connector
- JMS Connector

The RMI Connector is used by default and it is recommended that it be used at all times for monitoring and administrating the FioranoMQ server and/or its tools because of its easy, generic configuration and faster access compared to the JMS Connector. By default, the FioranoMQ RMI Connector component uses port 1858 for administration and monitoring and by default the Repeater RMI Connector uses port 1858. If two or more instances of the RMI Connector are running on the same system, the port number of all RMI Connectors must be unique.

11.2.10.1 To Change the RMI Port Number

To change the RMI Port number the steps listed below need to be followed:

1. Select the required profile from the **Select Profile Directory** dialog box.
2. Navigate to **%selectedProfile%>jmx>connector>RMIBasedJMXConnector** and change the port number from 1858 to the desired port number.
3. Once the changes are made in the **Properties Pane**, right-click on the %selectedProfile% in the **Profile Manager** and select the **Save** option. Changes are saved in the **Configs.xml** file.

11.2.10.2 To Monitor FioranoMQ Standalone Repeater using JMS Connector

To Monitor FioranoMQ Standalone Repeater using the JMS connector the steps listed below need to be followed:

1. Run the JMS Connector service with the standalone repeater. This can be done by adding **services\JMXConnector\JMS\JMXConnector1-service.xml** to the FMQRepeater.lst file.
2. The JMS Connector service running with the standalone repeater should connect to any remote FioranoMQ server in which no Connector service is currently running.
3. To monitor the FioranoMQ server with the repeater, launch Fiorano Studio using **%FIORANO_HOME%\Studio\bin\Studio.bat** (Studio.sh on UNIX Systems) or, if using Windows, through selecting the **Start > Programs > Fiorano > FioranoMQ > Fiorano Studio**
4. Connect to the FioranoMQ server using **Admin Studio**. The repeater node along with all the other server nodes will be displayed. The repeater node corresponds to the standalone repeater that is being run.
5. The standalone repeater may, therefore, be monitored online and various operations such as **add Link** or **remove Link** can be performed on the standalone repeater

11.3 Configuring/Monitoring Repeater in Online Mode

11.3.1 Online Configuration of Repeater Through Studio

For instructions regarding configuring and monitoring the repeater in the on-line mode refer to the section 23.7 Working with Repeater**.**

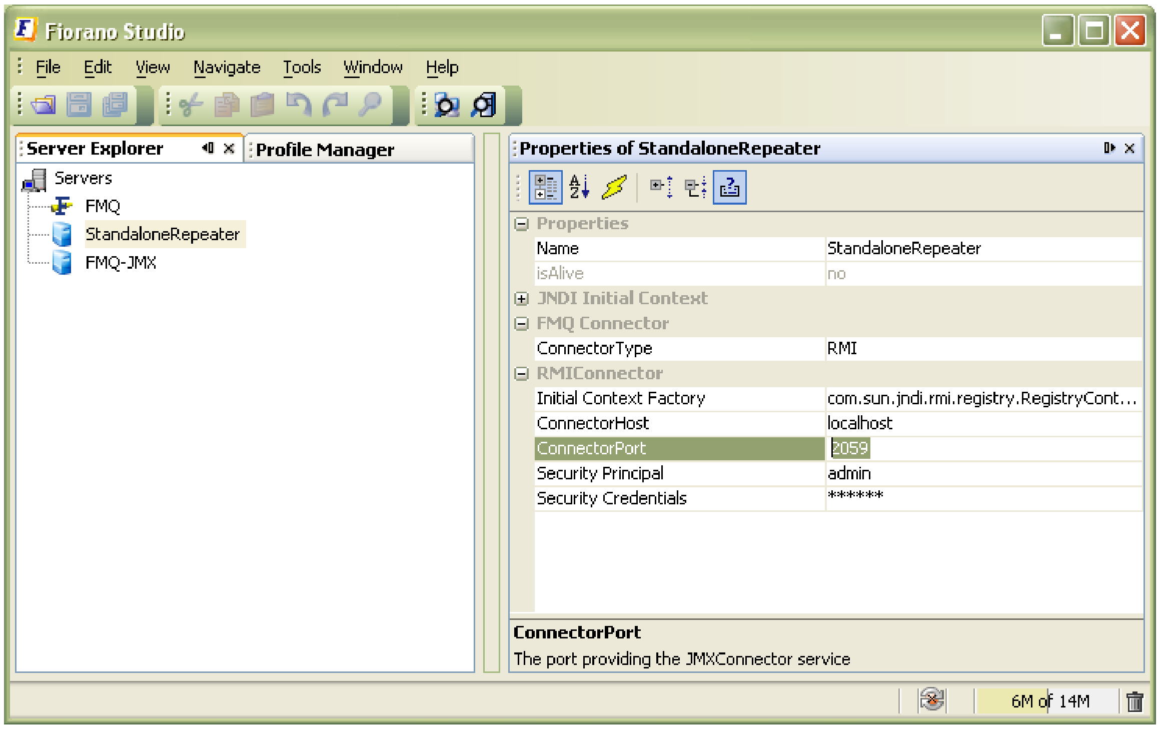
11.3.2 Online Configuration of Repeater Through JMX

This section describes all the configurations of Repeaters using the JMX connection.

11.3.2.1 Adding StandAloneRepeater node in the Server Explorer

To add a new StandAloneRepeater node, the steps listed below need to be followed:

1. From the **Server Explorer** pane, select the **FMQ-JMX** node. **Right-click** the mouse and click on the **Copy** option from the drop-down menu.
2. Select the **Servers** node and right-click the mouse and click on the **Paste** option.
3. Rename the new FMQ-JMX_1 to StandAloneRepeater.
4. In the properties window, change the RMIConnector's **ConnectorPort** to **2059**:



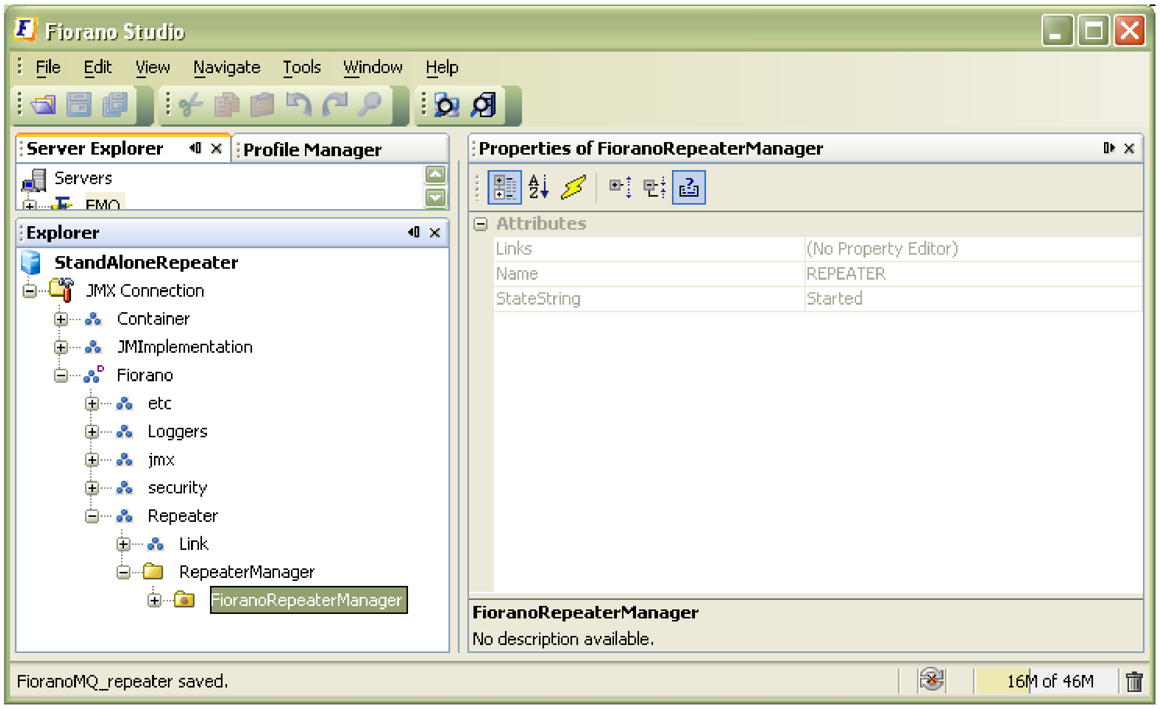
Properties	
Name	StandaloneRepeater
isAlive	no
JNDI Initial Context	
FMQ Connector	
ConnectorType	RMI
RMIConnector	
Initial Context Factory	com.sun.jndi.rmi.registry.RegistryCont...
ConnectorHost	localhost
ConnectorPort	2059
Security Principal	admin
Security Credentials	******

ConnectorPort
The port providing the JMXConnector service

11.3.2.2 Adding a Link

The administrator can create new replication links dynamically. This enables the applications to replicate messages on topics that are created after the repeater has started. Information such as the name of the link name, the source and target servers between the link, the protocol to be used for connection and login information, need to be provided when a new link is added to the repeater. To add a new link, follow the steps below:

1. Run the FMQ Server and run the StandAloneRepeater.
2. From the Server Explorer pane, login into StandAloneRepeater.
3. Navigate to Fiorano->Repeater->RepeaterManager->FioranoRepeaterManager and select the FioranoRepeaterManager node.



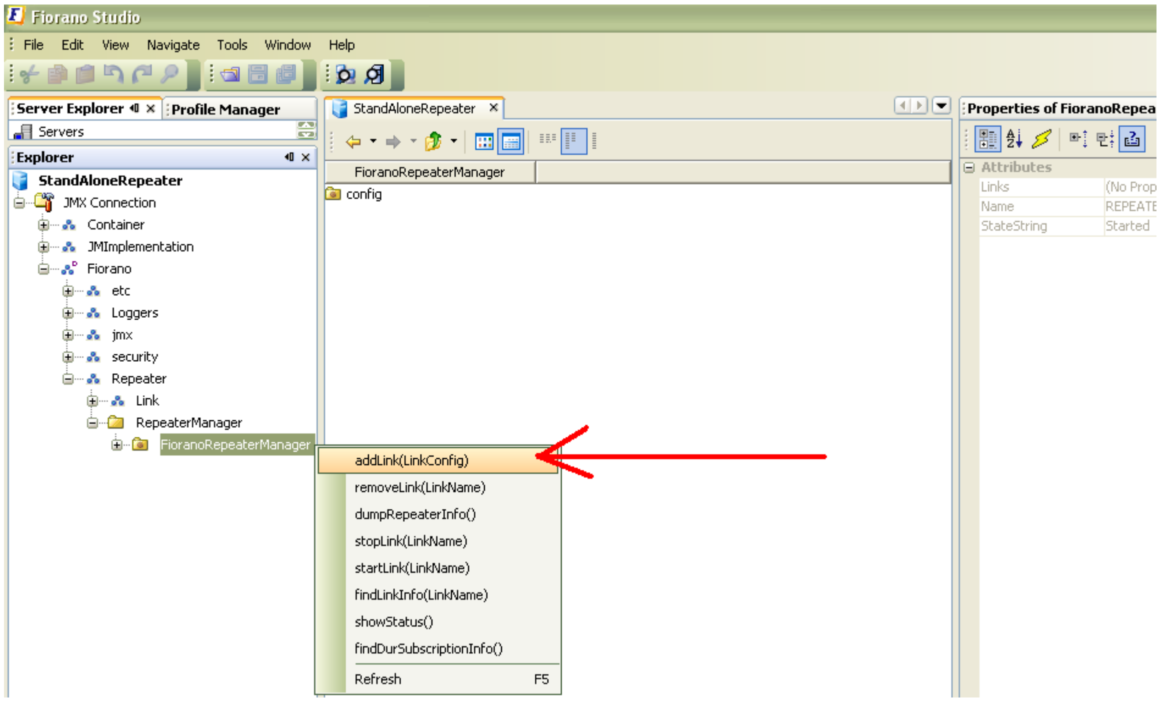
Links	(No Property Editor)
Name	REPEATER
StateString	Started

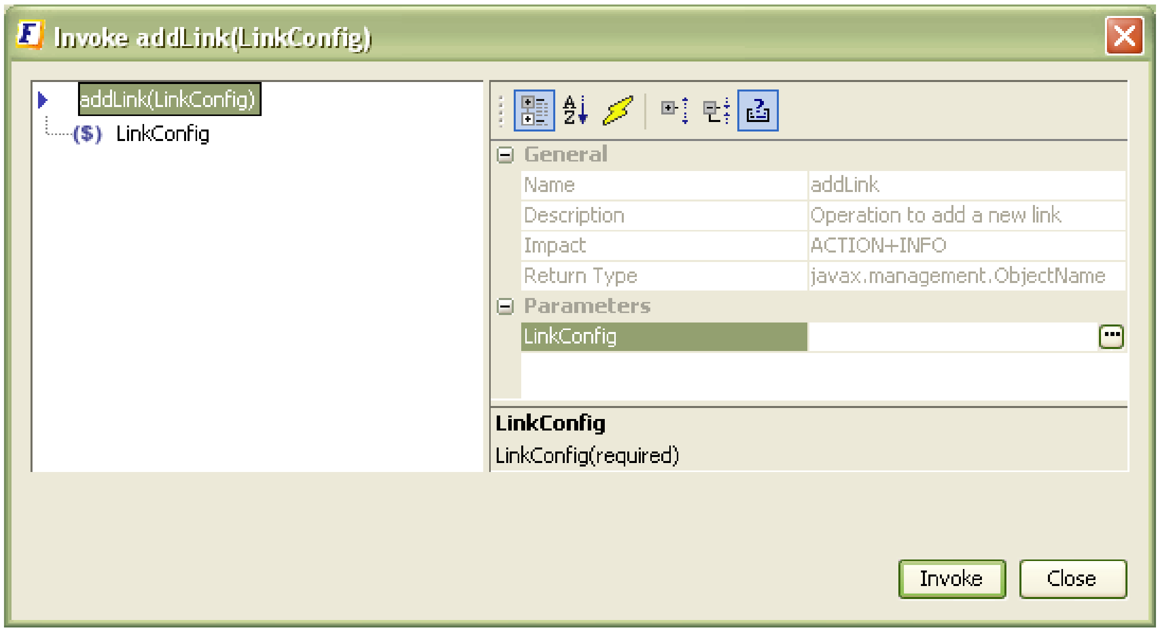
FioranoRepeaterManager
No description available.

FioranoMQ_repeater saved.

16M of 46M

4. Right-click the mouse and select the **addLink(LinkConfig)** from the drop-down menu. The **Invoke addLink(LinkConfig)** dialog box is displayed.

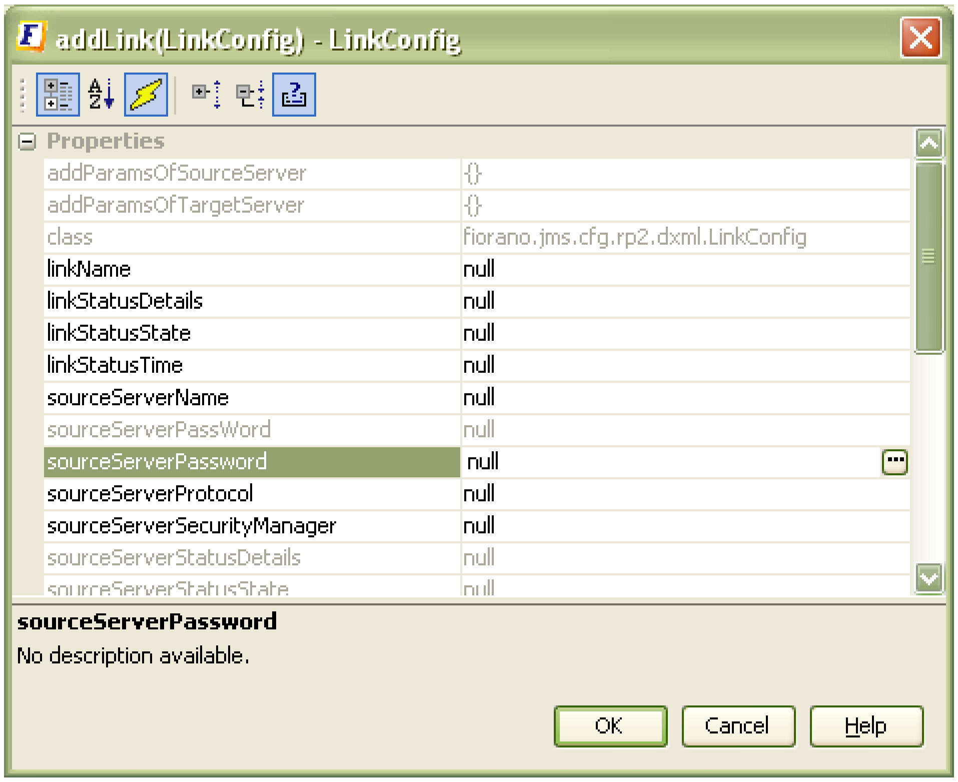




Name	addLink
Description	Operation to add a new link
Impact	ACTION+INFO
Return Type	javax.management.ObjectName
LinkConfig	

LinkConfig
LinkConfig(required)

5. Click the ⋯ **editor** button of parameter **LinkConfig**. The **addLink(LinkConfig) – LinkConfig** dialog box is displayed.



addParamsOfSourceServer	{}
addParamsOfTargetServer	{}
class	fiorano.jms.cfg.rp2.dxml.LinkConfig
linkName	null
linkStatusDetails	null
linkStatusState	null
linkStatusTime	null
sourceServerName	null
sourceServerPassWord	null
sourceServerPassword	null
sourceServerProtocol	null
sourceServerSecurityManager	null
sourceServerStatusDetails	null
sourceServerStatusState	null

sourceServerPassword

No description available.

6. Enter all required details. These include:

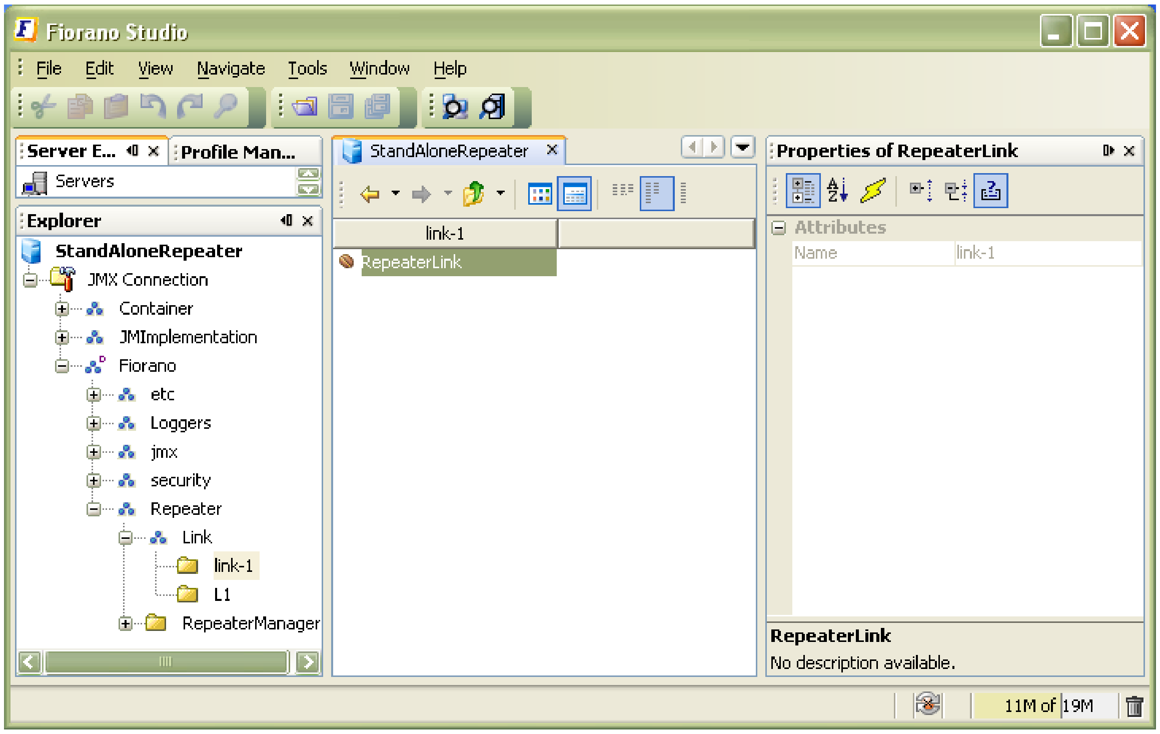
Parameter	**Default value**
linkName	Link-1
sourceServerName	FMQ
sourceServerPassword	anonymous
sourceServerProtocol	TCP
sourceServerSecurityManager	Repeater
sourceServerURL	http://localhost:1856
sourceServerUserName	anonymous
targetServerName	FMQ
targetServerPassword	anonymous
targetServerProtocol	TCP
targetServerSecurityManager	Repeater
targetServerURL	http://localhost:1856
targetServerUserName	anonymous

7. Click **OK** and **Invoke** the operation.
8. Close the **addLink** operation dialogue box.

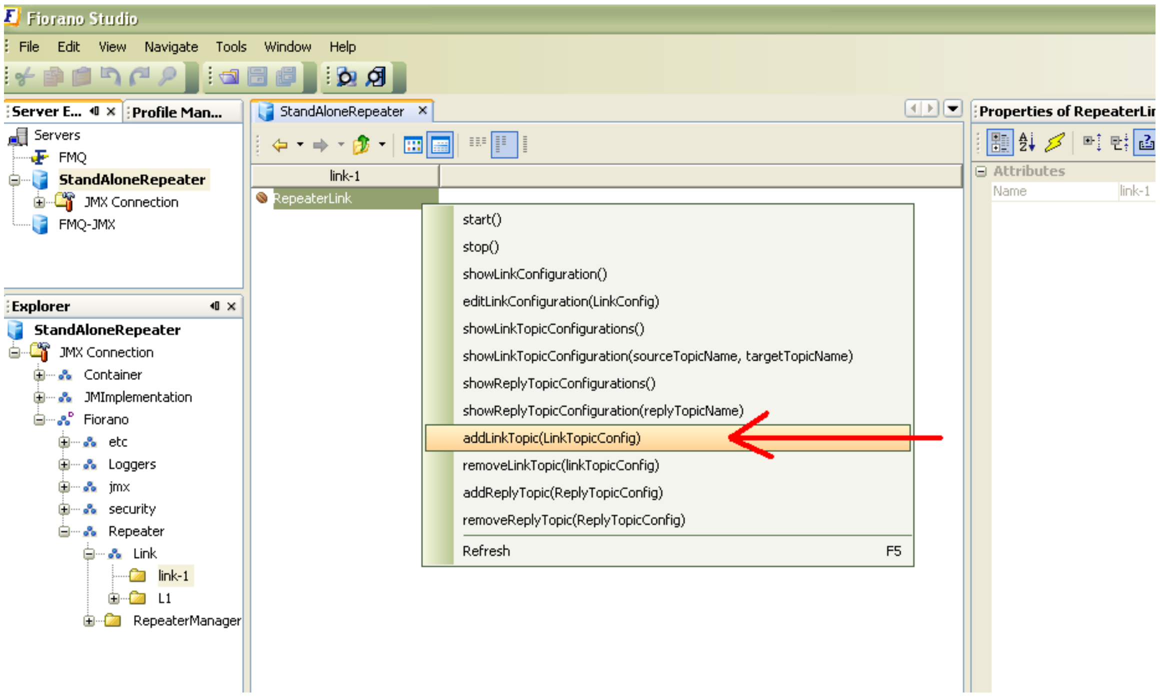
11.3.2.3 Adding a Link Topic

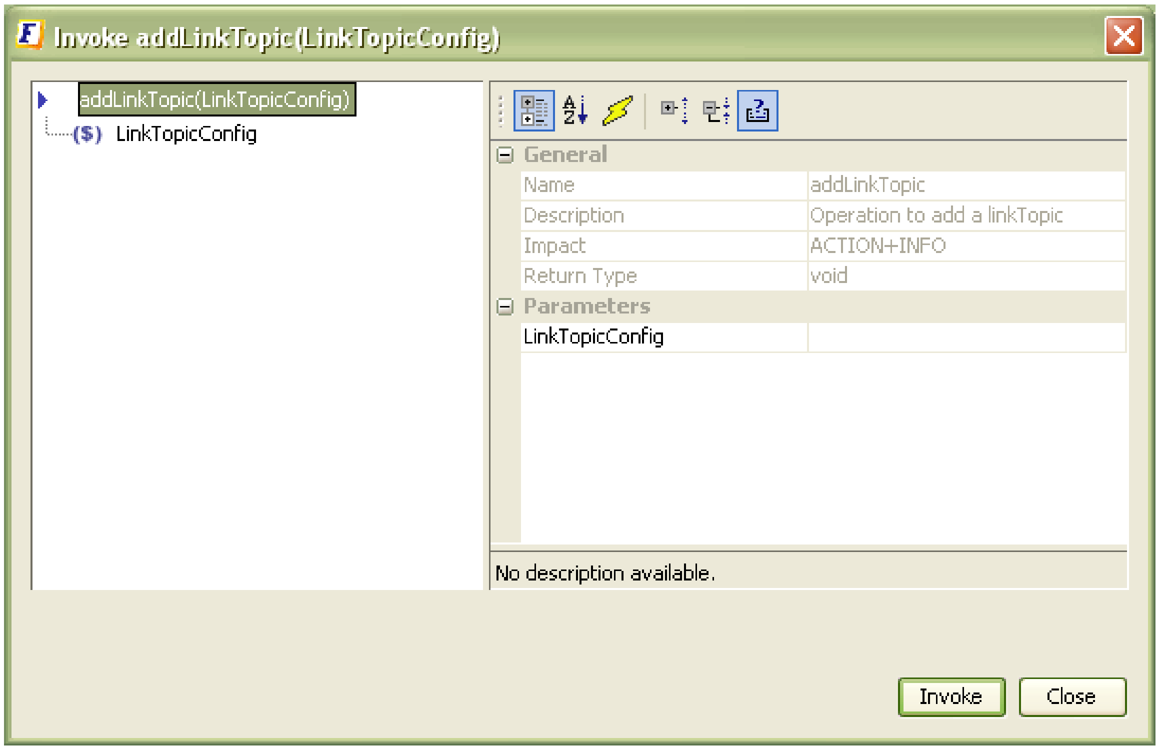
It is possible for the administrator to add one or more link topics to an existing topic. To add a link topic to an existing topic information such as the source topic name, the target topic name and the message selector need to be provided. To add a link topic follow the steps below::

1. Navigate to **Fiorano->Repeater->Link**, right-click the mouse and click the **Refresh** option. The link added is displayed.
2. Click upon the link displayed and select **RepeaterLink** MBean as shown in the figure below:



3. Right-click on the **RepeaterLink** MBean and select operation **addLinkTopic(LinkTopicConfig)**. The **Invoke addLinkTopic(LinkTopicConfig)** dialog box is displayed.

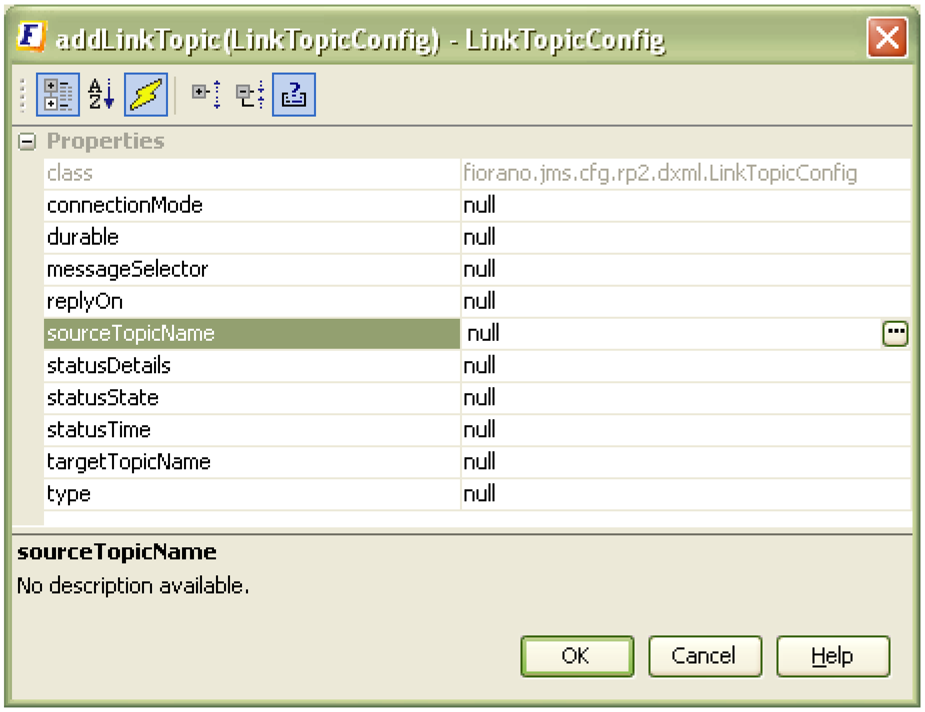

Fiorano Studio
File Edit View Navigate Tools Window Help
Server E... Profile Man...
Servers
FMQ
StandAloneRepeater
JMX Connection
FMQ-JMX
Explorer
StandAloneRepeater
JMX Connection
Container
JMImplementation
Fiorano
etc
Loggers
jmx
security
Repeater
Link
link-1
L1
RepeaterManager
StandAloneRepeater
link-1
RepeaterLink
start()
stop()
showLinkConfiguration()
editLinkConfiguration(LinkConfig)
showLinkTopicConfigurations()
showLinkTopicConfiguration(sourceTopicName, targetTopicName)
showReplyTopicConfigurations()
showReplyTopicConfiguration(replyTopicName)
addLinkTopic(LinkTopicConfig)
removeLinkTopic(linkTopicConfig)
addReplyTopic(ReplyTopicConfig)
removeReplyTopic(ReplyTopicConfig)
Refresh F5
Properties of RepeaterLi
Attributes
Name link-1


Invoke addLinkTopic(LinkTopicConfig)
addLinkTopic(LinkTopicConfig)
($) LinkTopicConfig
General
Parameters
Invoke
Close

Name	addLinkTopic
Description	Operation to add a linkTopic
Impact	ACTION+INFO
Return Type	void
LinkTopicConfig	

No description available.

4. Click editor button of parameter **LinkTopicConfig.** The **addLinkTopic(LinkTopicConfig) - LinkTopicConfig** dialog box is displayed .



class	fiorano.jms.cfg.rp2.dxml.LinkTopicConfig
connectionMode	null
durable	null
messageSelector	null
replyOn	null
sourceTopicName	null
statusDetails	null
statusState	null
statusTime	null
targetTopicName	null
type	null

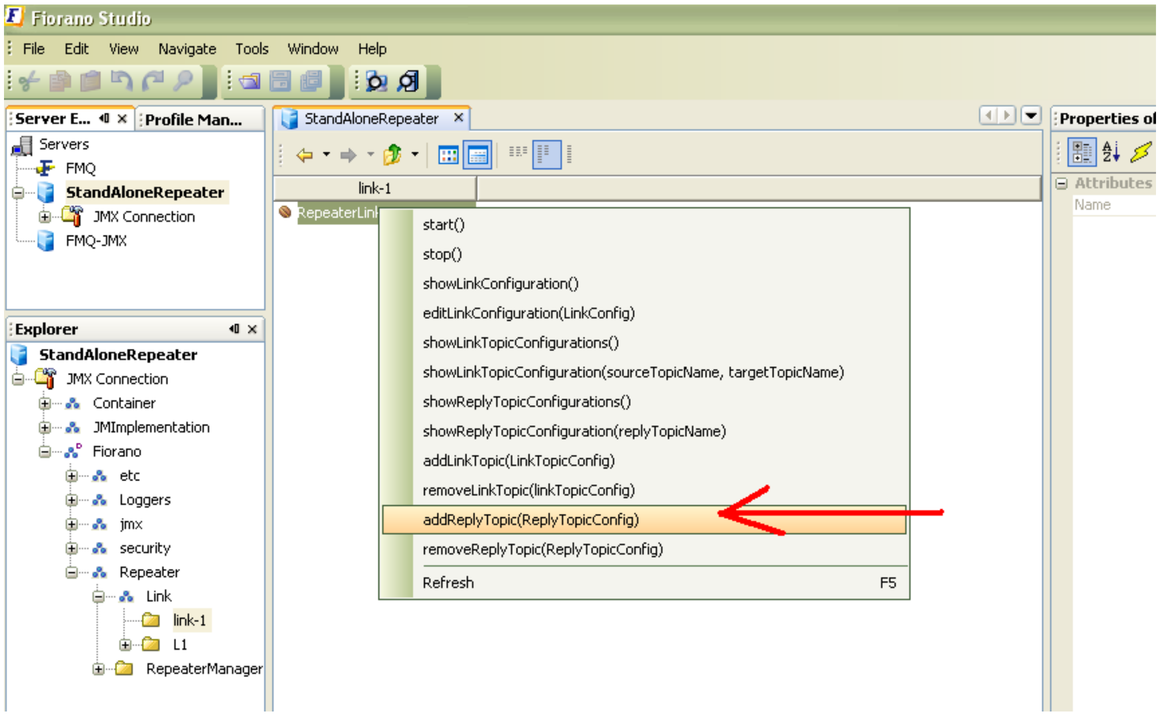
sourceTopicName
No description available.

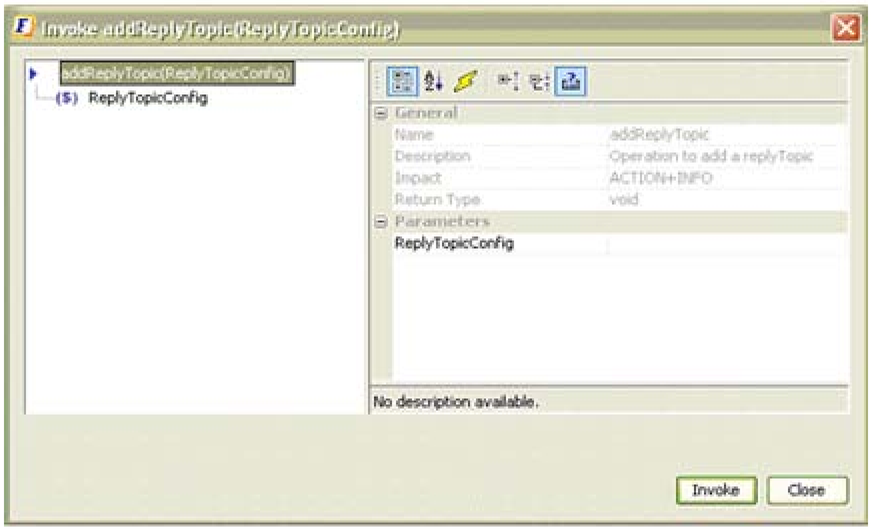
5. Fill out the required parameters. Click **OK** and the **Invoke**.

11.3.2.4 Adding a Reply Topic

It is possible for the administrator to add one or more reply topics to existing topics. To add one or more reply topics to existing topics information such as the source topic name, the target topic name and message selector need to be provided. To add a Reply topic, follow the steps below:

1. Navigate to Fiorano->Repeater->Link->(LinkName) and select theRepeaterLink MBean.
2. Right-click on the **RepeaterLink** MBean and select operation **addReplyTopic(ReplyTopicConfig)**. The **Invoke addReplyTopic(ReplyTopicConfig)** dialog box is displayed.

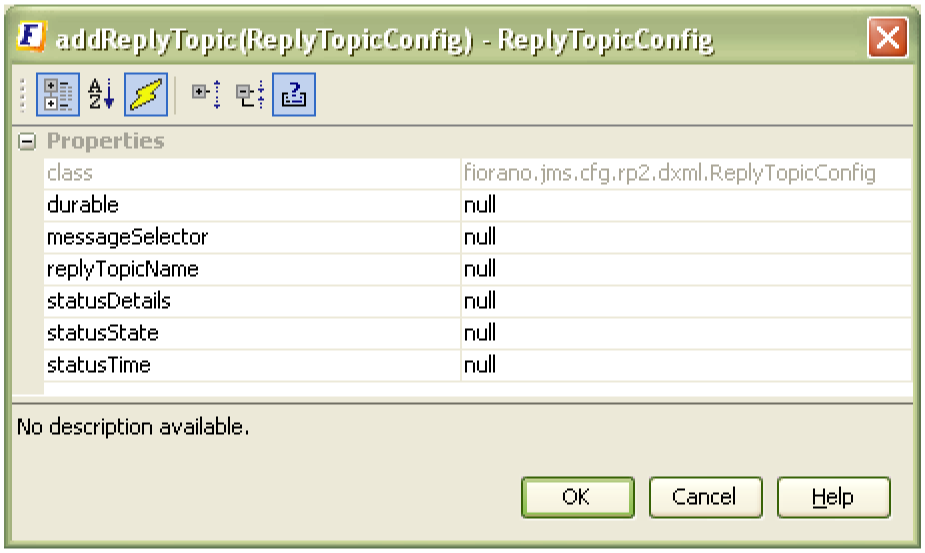

Fiorano Studio
File Edit View Navigate Tools Window Help
Server E... Profile Man...
Servers
FMQ
StandAloneRepeater
JMX Connection
FMQ-JMX
Explorer
StandAloneRepeater
JMX Connection
Container
JMImplementation
Fiorano
etc
Loggers
jmx
security
Repeater
Link
link-1
L1
RepeaterManager
StandAloneRepeater
link-1
RepeaterLink
start()
stop()
showLinkConfiguration()
editLinkConfiguration(LinkConfig)
showLinkTopicConfigurations()
showLinkTopicConfiguration(sourceTopicName, targetTopicName)
showReplyTopicConfigurations()
showReplyTopicConfiguration(replyTopicName)
addLinkTopic(LinkTopicConfig)
removeLinkTopic(linkTopicConfig)
addReplyTopic(ReplyTopicConfig)
removeReplyTopic(ReplyTopicConfig)
Refresh F5
Properties of
Attributes
Name


Invoke addReplyTopic(ReplyTopicConfig)
addReplyTopic(ReplyTopicConfig)
($) ReplyTopicConfig
General
Parameters
ReplyTopicConfig
Invoke
Close

Name	addReplyTopic
Description	Operation to add a replyTopic
Impact	ACTION+INFO
Return Type	void

No description available.

3. Click the editor button of the parameter ReplyTopicConfig. The addReplyTopic(ReplyTopicConfig)- ReplyTopicConfig dialog box is displayed.


addReplyTopic(ReplyTopicConfig) - ReplyTopicConfig

Properties

class	fiorano.jms.cfg.rp2.dxml.ReplyTopicConfig
durable	null
messageSelector	null
replyTopicName	null
statusDetails	null
statusState	null
statusTime	null

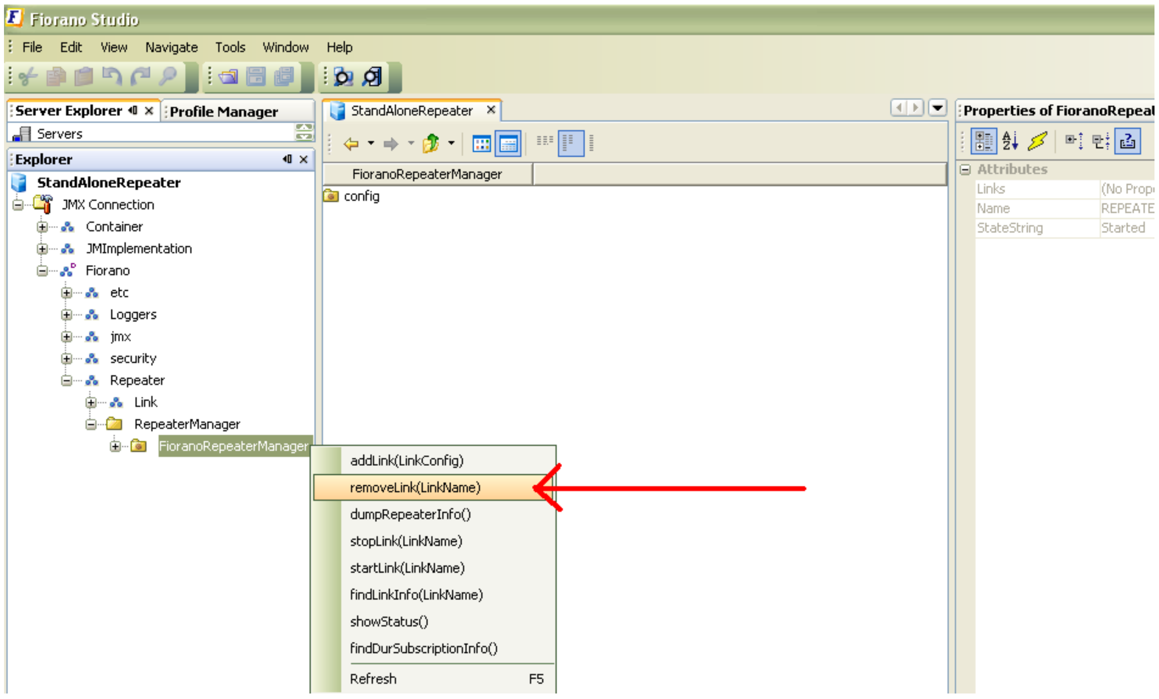
No description available.

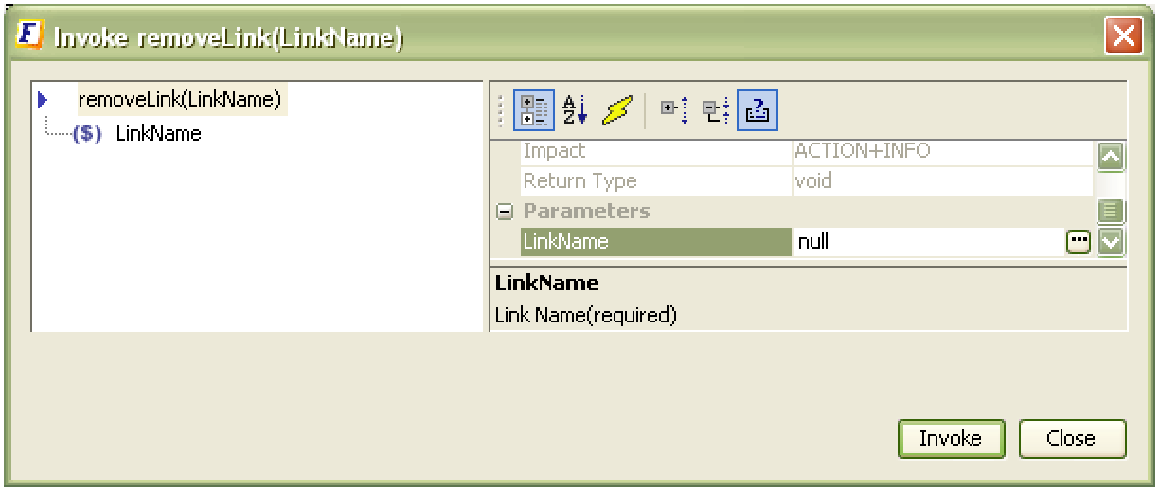
4. Specify the properties and click the **OK** button and **Invoke** the operation.

11.3.2.5 Removing a Link

The administrator needs to follow the steps below to remove links from a repeater:

1. Navigate to Fiorano->Repeater->RepeaterManager->FioranoRepeaterManager and select the FioranoRepeaterManager node.
2. Right-click on the **FioranoRepeaterManager** node and select operation **removeLink(LinkName)**. The **Invoke removeLink(LinkName)** dialog box is displayed.

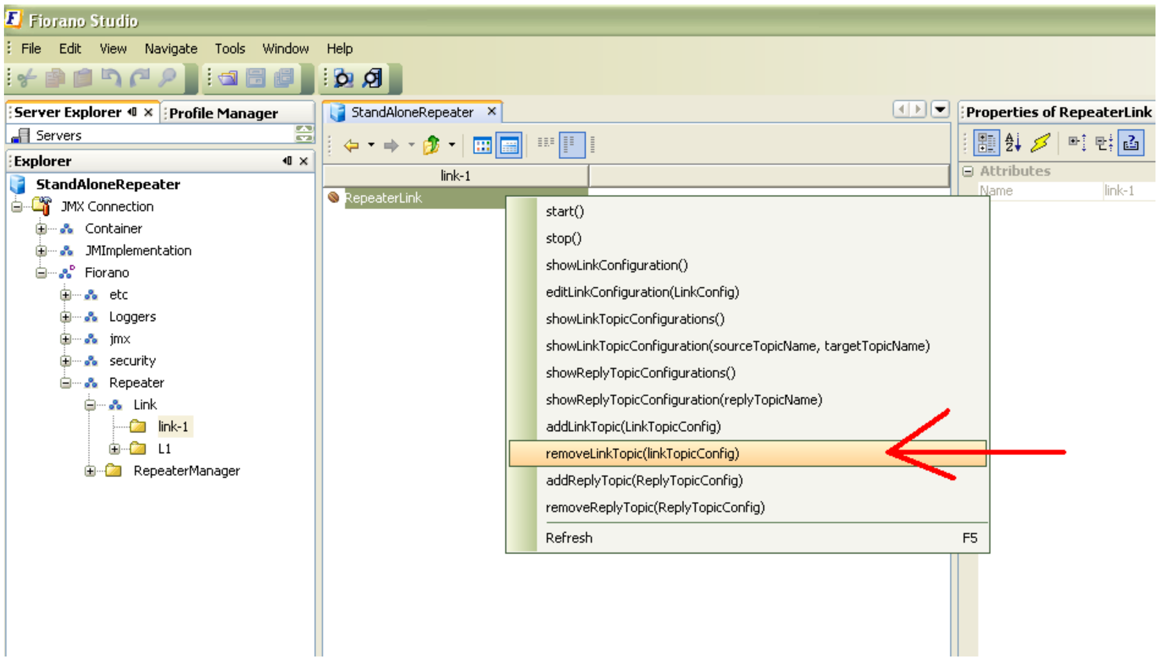




3. Specify the link name and click **Invoke**button.]

11.3.2.6 Removing a Link Topic

1. The administrator needs to follow the steps below to remove a link topic from the repeater. Navigate to Fiorano->Repeater->Link->(LinkName) and select the RepeaterLink MBean.
2. Right-click on the **RepeaterLink** MBean and select **removeLinkTopic(LinkTopicConfig)**.

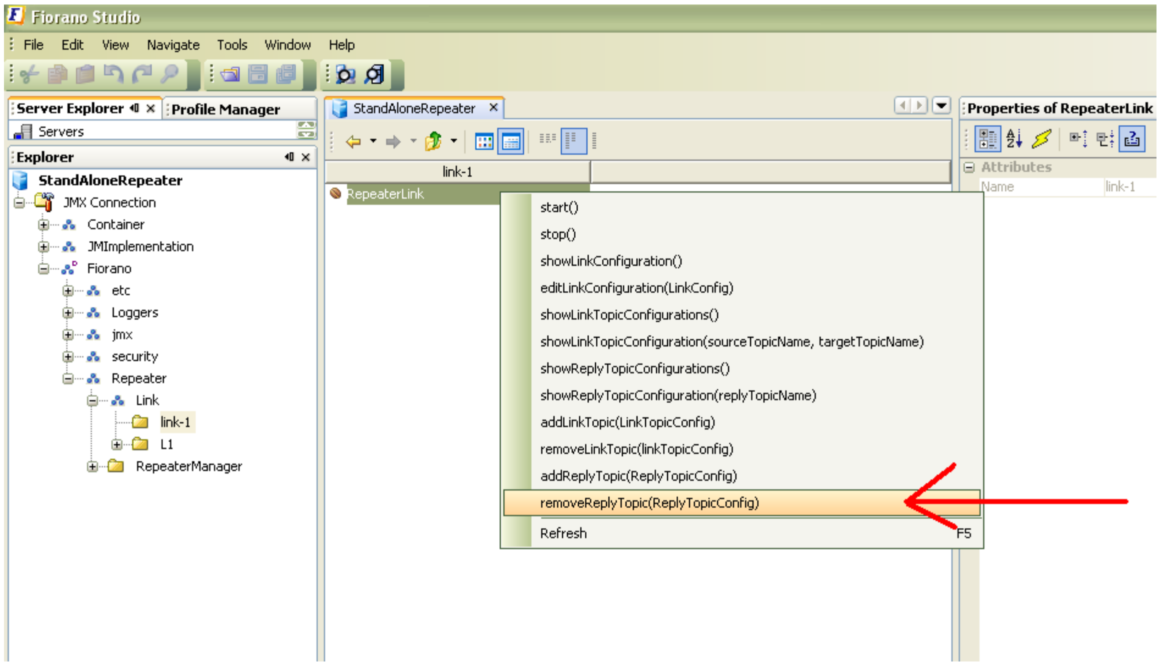

Fiorano Studio
File Edit View Navigate Tools Window Help
Server Explorer
Profile Manager
Servers
Explorer
StandAloneRepeater
JMX Connection
Container
JMImplementation
Fiorano
etc
Loggers
jmx
security
Repeater
Link
link-1
L1
RepeaterManager
StandAloneRepeater
link-1
RepeaterLink
start()
stop()
showLinkConfiguration()
editLinkConfiguration(LinkConfig)
showLinkTopicConfigurations()
showLinkTopicConfiguration(sourceTopicName, targetTopicName)
showReplyTopicConfigurations()
showReplyTopicConfiguration(replyTopicName)
addLinkTopic(LinkTopicConfig)
removeLinkTopic(linkTopicConfig)
addReplyTopic(ReplyTopicConfig)
removeReplyTopic(ReplyTopicConfig)
Refresh F5
Properties of RepeaterLink
Attributes
Name link-1

3. Other steps are similar to the **addLinkTopic**.

11.3.2.7 Removing a Reply Topic

The administrator needs to follow the steps below to remove the selected link topic from the repeater:

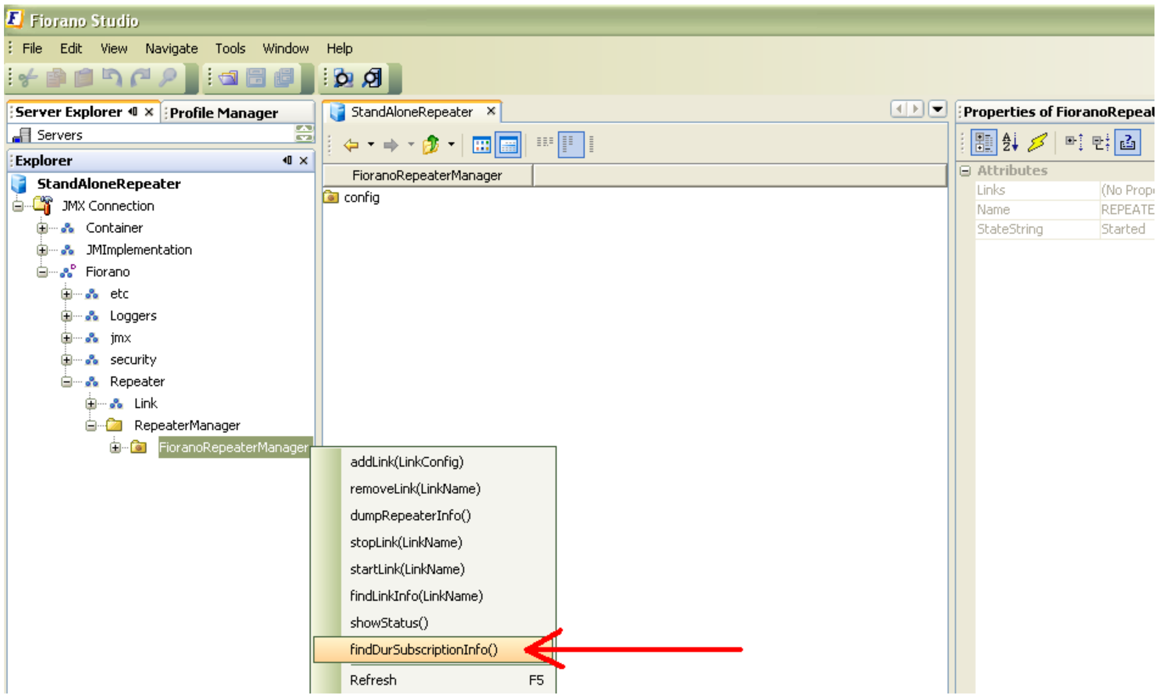
1. Navigate to **Fiorano->Repeater->Link->(LinkName)** and select the **RepeaterLink MBean**.
2. Right-click on the **RepeaterLink** MBean and select **removeReplyTopic(ReplyTopicConfig)**.


Fiorano Studio
File Edit View Navigate Tools Window Help
Server Explorer
Profile Manager
Servers
Explorer
StandAloneRepeater
JMX Connection
Container
JMImplementation
Fiorano
etc
Loggers
jmx
security
Repeater
Link
link-1
L1
RepeaterManager
StandAloneRepeater
link-1
RepeaterLink
start()
stop()
showLinkConfiguration()
editLinkConfiguration(LinkConfig)
showLinkTopicConfigurations()
showLinkTopicConfiguration(sourceTopicName, targetTopicName)
showReplyTopicConfigurations()
showReplyTopicConfiguration(replyTopicName)
addLinkTopic(LinkTopicConfig)
removeLinkTopic(linkTopicConfig)
addReplyTopic(ReplyTopicConfig)
removeReplyTopic(ReplyTopicConfig)
Refresh F5
Properties of RepeaterLink
Attributes
Name link-1

For additional steps please refer to **addReplyTopic**.

11.3.2.8 Viewing Durable Subscribers for a Repeater

1. Navigate to **Fiorano->Repeater->RepeaterManager->FioranoRepeaterManager** and select the **FioranoRepeaterManager** node.
2. Right-click on **FioranoRepeaterManager** node and select findDurSubscriptionInfo().


Fiorano Studio
File Edit View Navigate Tools Window Help
Server Explorer
Profile Manager
Servers
Explorer
StandAloneRepeater
JMX Connection
Container
JMImplementation
Fiorano
etc
Loggers
jmx
security
Repeater
Link
RepeaterManager
FioranoRepeaterManager
StandAloneRepeater
FioranoRepeaterManager
config
addLink(LinkConfig)
removeLink(LinkName)
dumpRepeaterInfo()
stopLink(LinkName)
startLink(LinkName)
findLinkInfo(LinkName)
showStatus()
findDurSubscriptionInfo()
Refresh F5
Properties of FioranoRepeat
Attributes
Links (No Prop
Name REPEATE
StateString Started

Chapter 12: Bridge

12.1 Launching Bridge in Stand-Alone Mode

To launch the Bridge in Stand-Alone Mode follow the stepsbelow:

1. Openthe FioranoMQ console.
2. Navigate to %FIORANO_HOME%\fmq\bin directory.

A ready-to-use batch file to launch the Bridge is now available.

To run StandaloneBridge, follow the commands below:

fmq.bat -profile StandAloneBridge

On UNIX, use

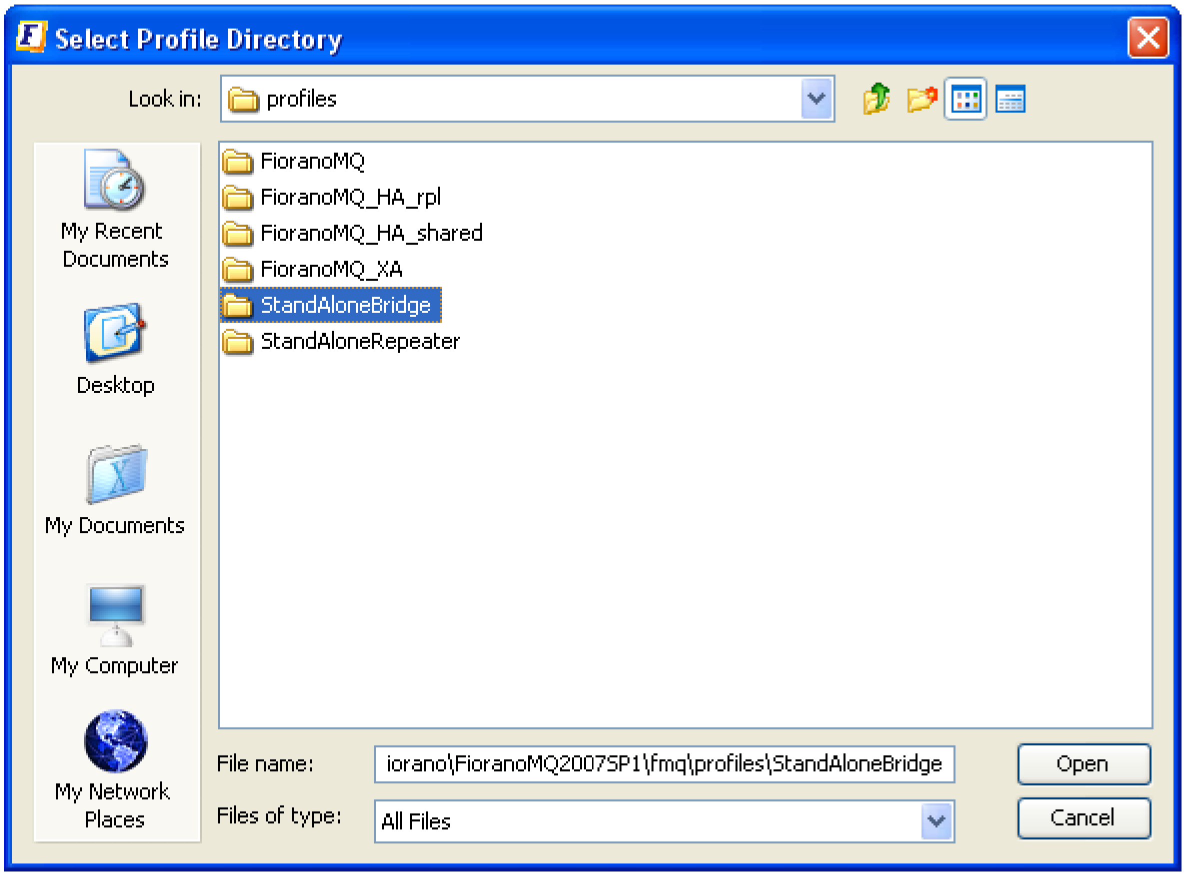
fmq.sh -profile StandAloneBridge

12.2 Configuring Bridge in Off-line Mode

In offline mode, the administrator is able to add links to the Bridge and configure the same to source and target servers for message replication. Cluster administrators are provided with a template configuration file (**Configs.xml** located in %FIORANO_HOME%\fmq\profiles\StandAloneBridge\conf directory of the FioranoMQ installation package) to simplify the Bridge configuration. This provides the default Bridge configuration with two links, thus linking the source and the target servers bi-directionally. This file must be renamed Configs.xml before starting Fiorano Admin Studio in offline mode. The tool displays the Bridge with the default links. The links can be configured using the steps below:

For information about configuring profiles through a text based file, refer to FioranoMQ Getting Started.

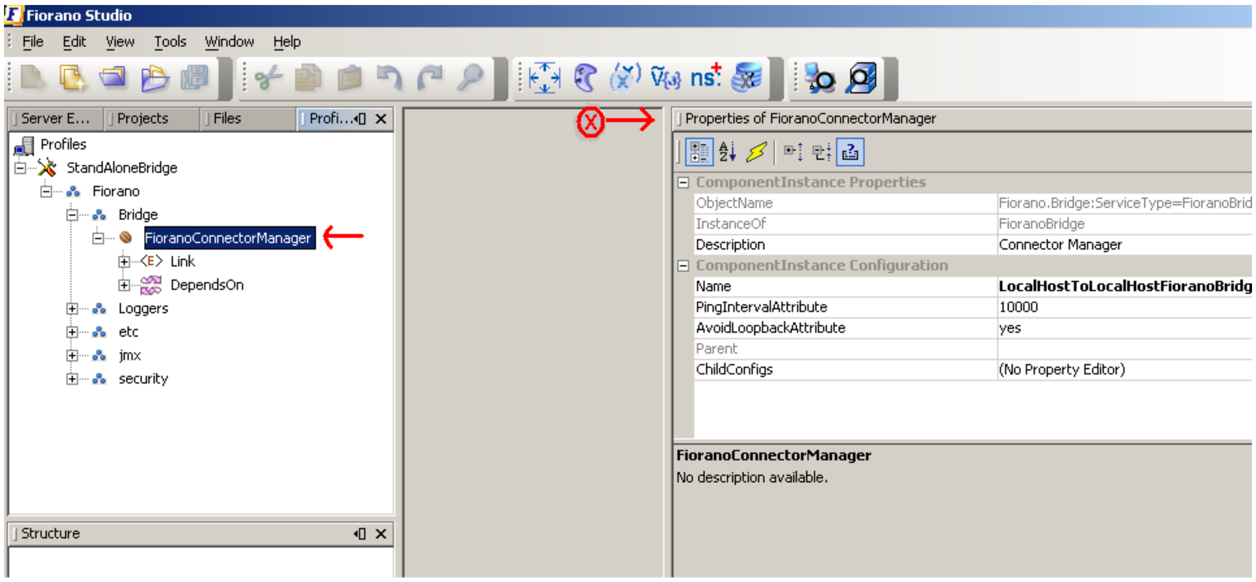
1. For offline configuration of the FioranoMQ server, launch Fiorano Studio using **%FIORANO_HOME%\Studio\bin\Studio.bat (**Use **/Studio.sh** if using UNIX). If using Windows, select **Start** > **Programs** > **Fiorano** > **FioranoMQ** > **Fiorano Studio > Fiorano Studio**
2. Select **Tools** > **Configure Profile** from the menu bar, and select the **StandAloneBridge** folder from the **Select Profile Directory** dialog box and click the **Open** button. The **StandAloneBridge** profile is opened in the **Profile Manager** pane.



FioranoMQ
FioranoMQ_HA_rpl
FioranoMQ_HA_shared
FioranoMQ_XA
StandAloneBridge
StandAloneRepeater

File name: iorano\FioranoMQ2007SP1\fmq\profiles\StandAloneBridge
Files of type: All Files

3. Navigate to **StandAloneBridge > Bridge > FioranoConnectorManager** node in the Server Explorer. The properties of the Bridge are displayed in the **Properties Pane**. Configure these as required.



ComponentInstance Properties	
ObjectName	Fiorano.Bridge:ServiceType=FioranoBrid
InstanceOf	FioranoBridge
Description	Connector Manager
ComponentInstance Configuration	
Name	LocalHostToLocalHostFioranoBridg
PingIntervalAttribute	10000
AvoidLoopbackAttribute	yes
Parent	
ChildConfigs	(No Property Editor)

FioranoConnectorManager
No description available.

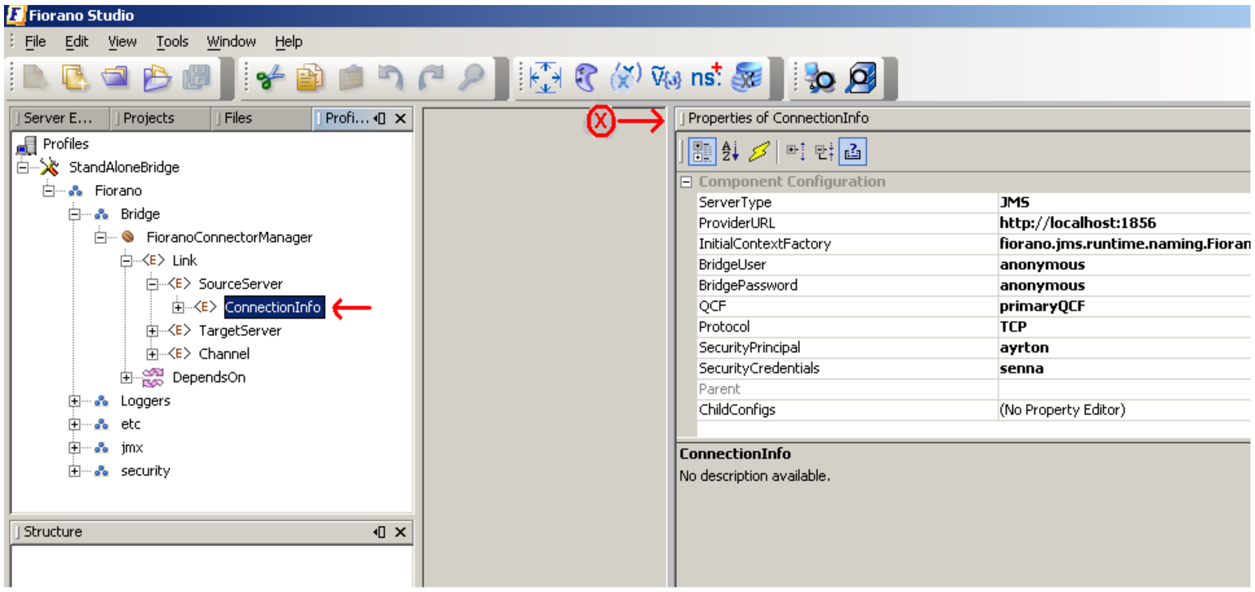
12.2.1 Editing a Link

Properties related to the default link can be edited before creating and managing additional links in the online mode. The Link element within the Bridge Manager MBean contains theelements listed below:

1. **SourceServer** - Specifies the server on which subscriptions are created. The Source Server element contains the ConnectionInfo.
2. **TargetServer** - Specifies the Server on which publishers are created. The TargetServer contains the ConnectionInfo.

To edit follow the steps below:

1. Navigate to the desired Link and expand its components.
2. Selecting one of the components causes its properties to be displayed in the Properties Pane (marked **X** in the figure). Properties can now be edited.



ServerType	JMS
ProviderURL	http://localhost:1856
InitialContextFactory	fiorano.jms.runtime.naming.Fioran
BridgeUser	anonymous
BridgePassword	anonymous
QCF	primaryQCF
Protocol	TCP
SecurityPrincipal	ayrton
SecurityCredentials	senna
Parent	
ChildConfigs	(No Property Editor)

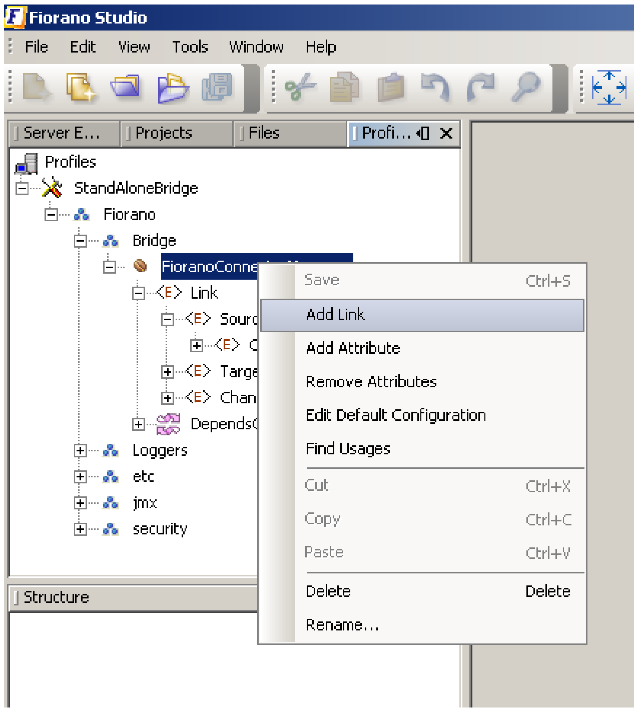
ConnectionInfo
No description available.

3. Save the configuration for the changes to take effect.

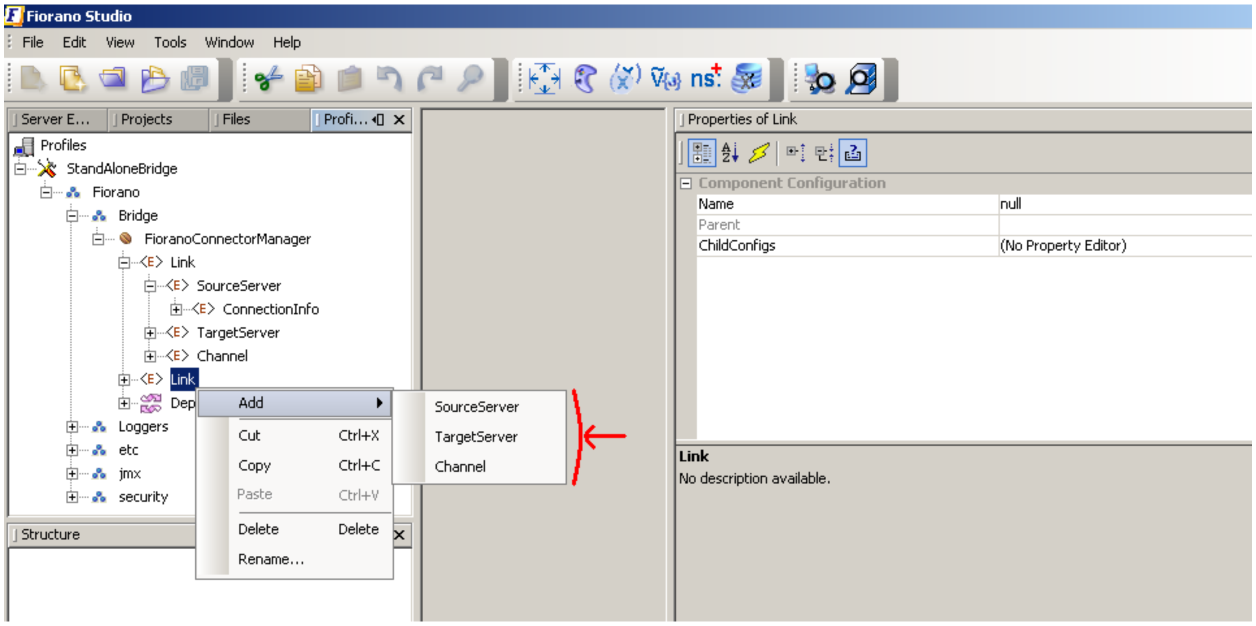
12.2.2 Adding a Link

The Bridge sends messages on the link specified between a source and a target server. A Bridge can have a number of links configured. By default, the server sets up only a single link to the Bridge. A new link can be added to the Bridge when it is configured in the offline mode, by following the steps below:

1. Navigate to **StandAloneBridge > Bridge > FioranoConnectorManager** node in the **Server Explorer**. Right-click the **FioranoConnectorManager** node and select the **Add Link** option from the pop-up menu. A new Link is added to the **FioranoConnectorManager**.



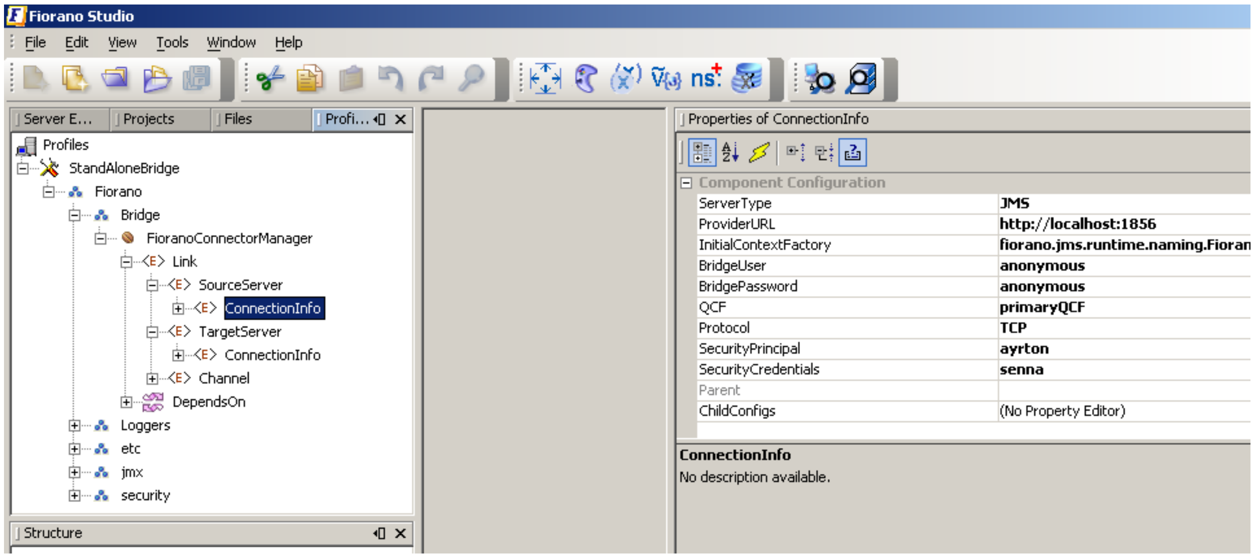
2. Now add other parameters such as SourceServer, TargetServer and Channel to this new Link.



Name	null
Parent	
ChildConfigs	(No Property Editor)

Link
No description available.

3. Configure SourceServer and TargetServer by adding **ConnectionInfo** (right-click over the node and select **Add ConnectionInfo** from the pop-up menu) and setting its properties, as shown in the figure below.



ServerType	JMS
ProviderURL	http://localhost:1856
InitialContextFactory	fiorano.jms.runtime.naming.Fioran
BridgeUser	anonymous
BridgePassword	anonymous
QCF	primaryQCF
Protocol	TCP
SecurityPrincipal	ayrton
SecurityCredentials	senna
Parent	
ChildConfigs	(No Property Editor)

ConnectionInfo
No description available.

4. Configure the Channel by adding **SrcQueue and TargetQueue** (right-click over the node and select **Add > SrcQueue/TargetQueue** from the pop-up menu) and set its properties.
5. Save the new configuration to render it effect and embed this into Configs.xml file.

12.2.3 Deleting a Link

Unwanted Links can be deleted by following the steps below:

1. Select the Link to be deleted..

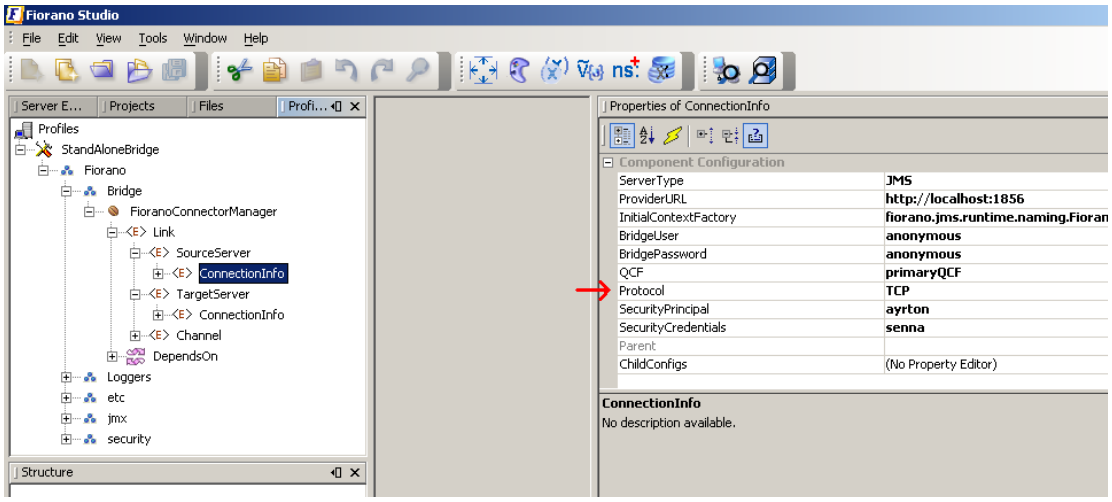
2. Right-click and select **Delete** from the shortcut menu. A **Confirm Object Deletion** dialog box is displayed. Click on **Yes** to delete the link.

12.2.4 Running a Bridge on Secure Protocol

FioranoMQ supports running the server on secure protocols. A Bridge can be run on a secure protocol to connect to a FioranoMQ server that runs on secure protocols. The parameters listed below need to be configured to run a FioranoMQ Bridge on protocols with security.

The Protocol Type of Source/Target servers must be the same as the protocol on which the Server is running. Possible values are:

- TCP with no security
- HTTP with no security
- SUN_SSL TCP with JSSE security
- HTTPS_SUN HTTP with JSSE security


Fiorano Studio
File Edit View Tools Window Help
Server E... Projects Files Profi...
Structure
Properties of ConnectionInfo

```
Profiles
  StandAloneBridge
    Fiorano
      Bridge
        FioranoConnectorManager
          <E> Link
            <E> SourceServer
              <E> ConnectionInfo
            <E> TargetServer
              <E> ConnectionInfo
            <E> Channel
          DependsOn
      Loggers
      etc
      jmx
      security
```

Component Configuration	
ServerType	JMS
ProviderURL	http://localhost:1856
InitialContextFactory	fiorano.jms.runtime.naming.Fioran
BridgeUser	anonymous
BridgePassword	anonymous
QCF	primaryQCF
Protocol	TCP
SecurityPrincipal	ayrton
SecurityCredentials	senna
Parent	
ChildConfigs	(No Property Editor)

ConnectionInfo
No description available.

ServerSecurityManager must be set to the name of the class that is used for authenticating the client with the server. By default, FioranoMQ runtime provides one implementation for each JSSE enabled SSLs.

fiorano.jms.runtime.sm.JSSESecurityManager for JSSE.

Change properties in the Bridge, Save all changes.

The security enabled Bridge should be able to runs and connect to a server running on a secure protocol in order to replicate messages.

12.2.5 Configuring Monitoring Support

FioranoMQ comes with two different types of Connector mechanisms that can be used to monitor and administer FioranoMQ and/or its tools like the dispatcher/bridge/repeater etc. The Connectors are:

- RMI Connector
- JMS Connector

By default the RMI Connector is used. It is recommend that the RMI Connector always be used for monitoring and administrating the FioranoMQ server or its tools given its generic configuration and fast access, as compared to the JMS Connector. By default, the FioranoMQ RMI Connector component uses port 1858 for administrating and monitoring while the Bridge RMI Connector uses port 1899. If two or more instances of the RMI Connector are running on the same system, the port number of each RMI Connectors must be unique.

12.2.5.1 To Change the RMI port number of a Profile

To change the RMI port number of a profile follow the steps below:

1. Select the required profile from the **Select Profile Directory** dialog box.
2. Navigate to **%selectedProfile%>jmx>connector>RMIBasedJMXConnector** and change the port number from 1858 to the required value.
3. From the **Properties Pane**, right-click on the **%selectedProfile%** in the **Profile Manager.** Select the **Save** option from the menu that is displayed. Changes get saved in the **Configs.xml** file.

12.2.5.2 To Monitor the FioranoMQ Standalone Bridge using the JMS Connector

To monitor the FioranoMQ Standalone Bridge using the JMS Connector follow the steps below:

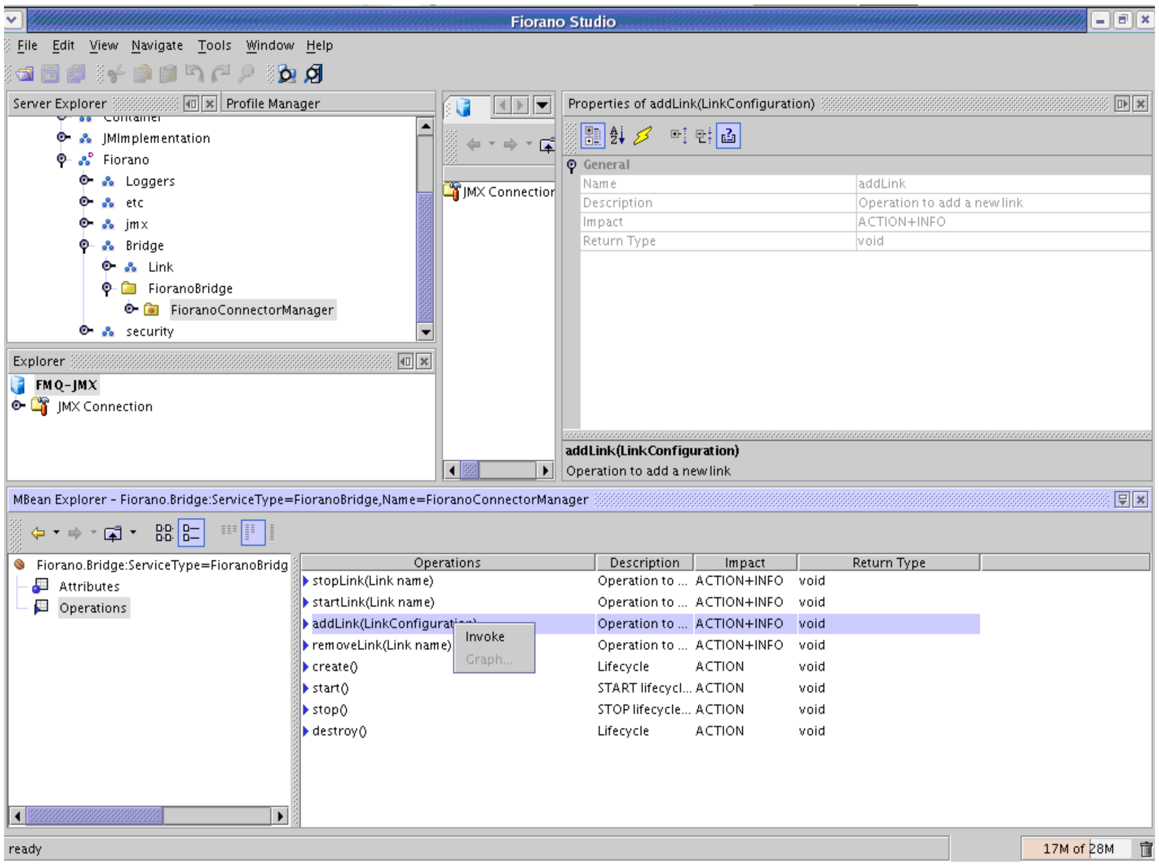
1. Run the JMS Connector service with the Standalone Bridge. This can be done by adding **services\JMXConnector\JMS\JMXConnector1-service.xml** to the **FMQRepeater.lst** file.
2. The JMS Connector Service running with the standalone bridge is able to connect to any remote FioranoMQ server on which no other Connector service is running.
3. To monitor the FioranoMQ server with the bridge, launch Fiorano Studio using **%FIORANO_HOME%\Studio\bin\Studio.bat** (use /Studio.sh if using UNIX).If using Windows, select **Start > Programs > Fiorano > FioranoMQ > Fiorano Studio**.
4. Connect to the FioranoMQ server using **Admin Studio**. The Bridge node along with all the other server nodes is displayed. The Bridge node should correspond to the Standalone Bridge that running.
    - The standalone Bridge can, thus, be monitored online and various operations like add Link, remove Link can be performed on the Bridge.

12.3 Configuring Bridge in Online Mode

The Bridge can be configured in two ways though the online mode. All added configurations work as long as the Bridge is running. Once the Bridge is stopped, all configurations are lost.

12.3.1 Configuring Through FMQ-JMX Login

1. To configure the Bridge online, login to the Bridge through **FMQ-JMX** from the Studio. The User can perform a RMI based JMX login to the Bridge from Studio by providing the RMI Port. After a **FMQ-JMX** login from Studio, Navigate to **JMX Connection-->Fiorano-->Bridge-->FioranoConnectorManager**.

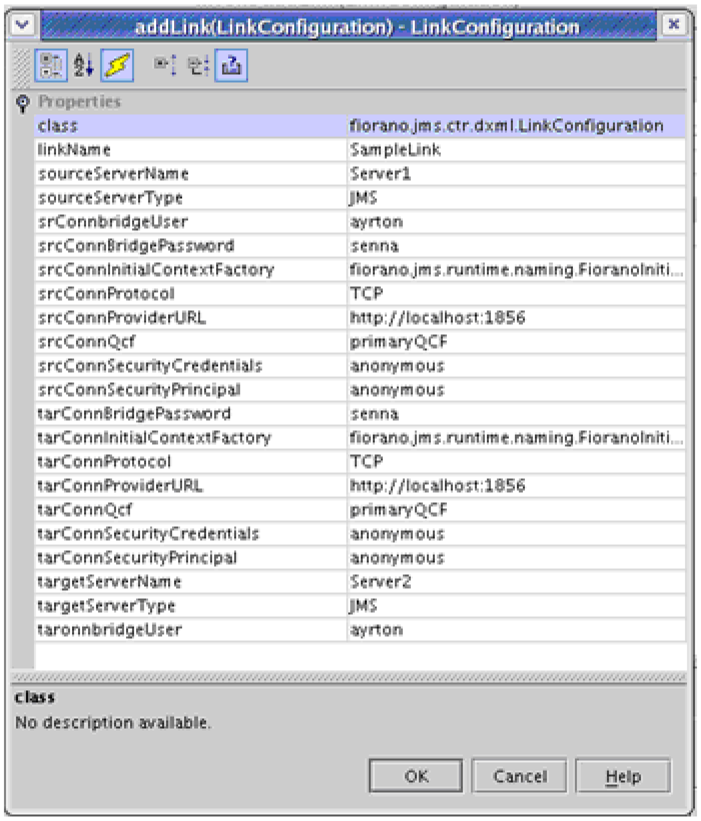


Name	addLink
Description	Operation to add a new link
Impact	ACTION+INFO
Return Type	void

addLink(LinkConfiguration)
Operation to add a new link

Operations	Description	Impact	Return Type
stopLink(Link name)	Operation to ...	ACTION+INFO	void
startLink(Link name)	Operation to ...	ACTION+INFO	void
addLink(LinkConfiguration)	Operation to ...	ACTION+INFO	void
removeLink(Link name)	Operation to ...	ACTION+INFO	void
create()	Lifecycle	ACTION	void
start()	START lifecycl...	ACTION	void
stop()	STOP lifecycle...	ACTION	void
destroy()	Lifecycle	ACTION	void

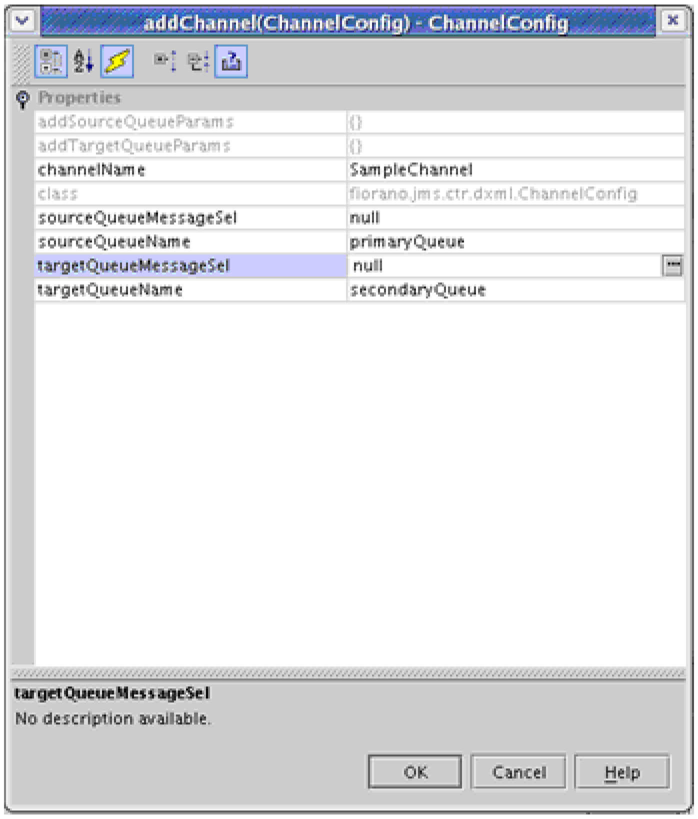
2. Right-click the **FioranoConnectionManager** and select the **addLink(LinkConfiguration)** option from the pop-up menu. A **LinkConfiguration** dialog box is displayed, as shown in the figure below.
3. Invoke the **addLink** operation by assigning appropriate values to **LinkConfiguration**.



class	fiorano.jms.ctr.dxml.LinkConfiguration
linkName	SampleLink
sourceServerName	Server1
sourceServerType	JMS
srConnbridgeUser	ayrton
srcConnBridgePassword	senna
srcConnInitialContextFactory	fiorano.jms.runtime.naming.FioranoIniti...
srcConnProtocol	TCP
srcConnProviderURL	http://localhost:1856
srcConnQcf	primaryQCF
srcConnSecurityCredentials	anonymous
srcConnSecurityPrincipal	anonymous
tarConnBridgePassword	senna
tarConnInitialContextFactory	fiorano.jms.runtime.naming.FioranoIniti...
tarConnProtocol	TCP
tarConnProviderURL	http://localhost:1856
tarConnQcf	primaryQCF
tarConnSecurityCredentials	anonymous
tarConnSecurityPrincipal	anonymous
targetServerName	Server2
targetServerType	JMS
taronnbridgeUser	ayrton

class
No description available.

4. Once theLink is created, navigate to **JMX Connection-->Fiorano-->Bridge-->Link-->LinkName**. The LinkName is provided by the User when creating the link. (If the Link runtime MBean is not displayed via Studio, right-click on **Link** to refresh it). Perform the **addChannel** operation by assigning appropriate values in the configuration wizard.



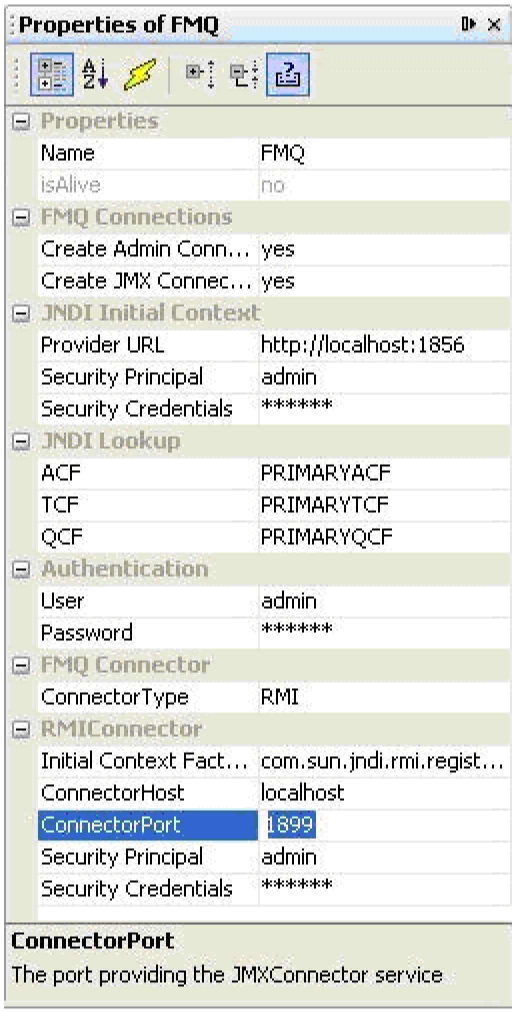
addSourceQueueParams	{}
addTargetQueueParams	{}
channelName	SampleChannel
class	fiorano.jms.ctr.dxml.ChannelConfig
sourceQueueMessageSel	null
sourceQueueName	primaryQueue
targetQueueMessageSel	null
targetQueueName	secondaryQueue

targetQueueMessageSel
No description available.

5. The **add channel** ‘function’ is visible to the user at the end of performing the add channel operation. A User can modify the link configuration by performing the **editLinkConfiguration operation**. All offline operations can be performed here while corresponding operations are performed at appropriate locations.

12.3.2 Configuring Through the FMQ Login

1. To configure the StandAloneBridge through the **FMQ** login, change the RMI **ConnectorPort** to the port on which the Bridge is running. By default, the **ConnectorPort** is **1858**. Change the **ConnectorPort** to **1899** for StandAloneBridge. Log into the **FMQ** server through Studio.

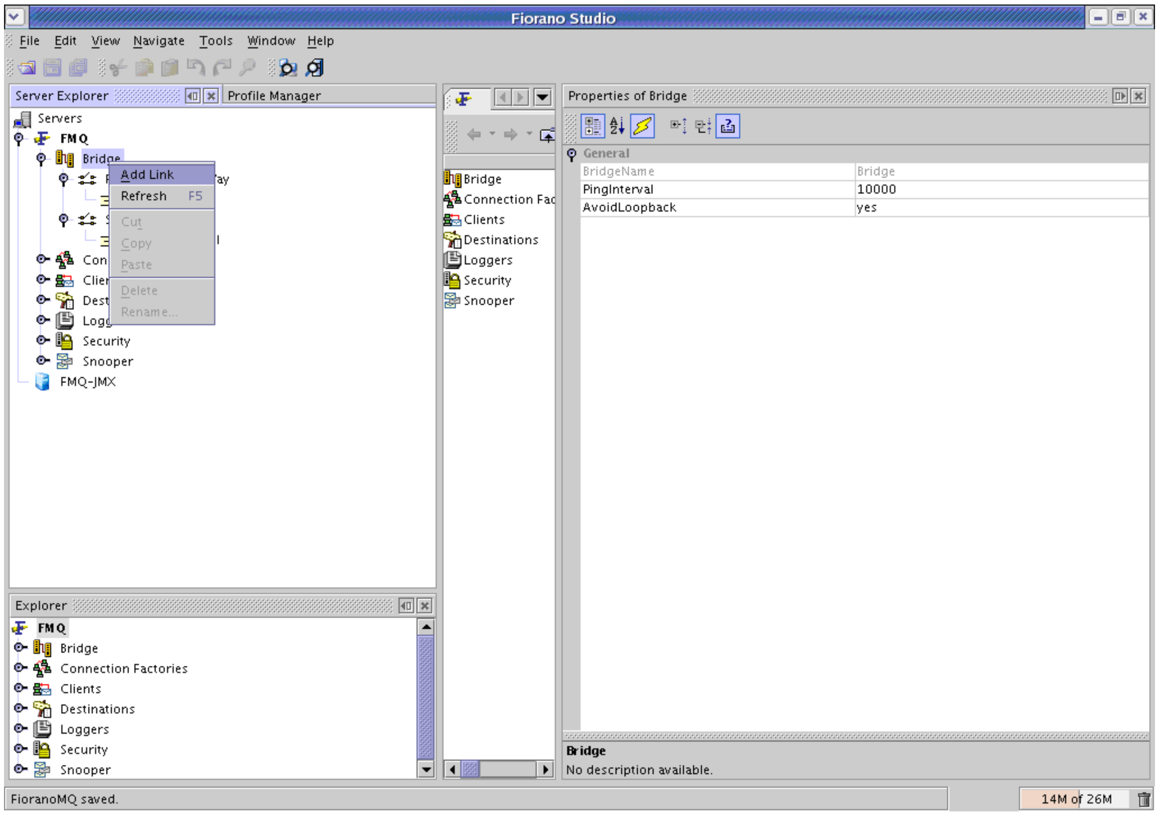


Properties of FMQ

Properties	
Name	FMQ
isAlive	no
FMQ Connections	
Create Admin Conn...	yes
Create JMX Connec...	yes
JNDI Initial Context	
Provider URL	http://localhost:1856
Security Principal	admin
Security Credentials	******
JNDI Lookup	
ACF	PRIMARYACF
TCF	PRIMARYTCF
QCF	PRIMARYQCF
Authentication	
User	admin
Password	******
FMQ Connector	
ConnectorType	RMI
RMIConnector	
Initial Context Fact...	com.sun.jndi.rmi.regist...
ConnectorHost	localhost
ConnectorPort	1899
Security Principal	admin
Security Credentials	******

ConnectorPort
The port providing the JMXConnector service

2. After logging into the **FMQ** server, right-click on the **Bridge** and click the **Add Link** option from the pop-up menu. The **New Link Properties** dialog box is displayed.

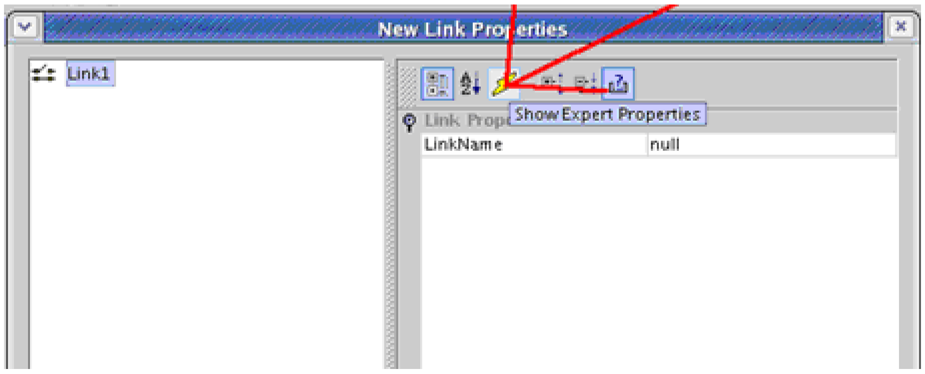


BridgeName	Bridge
PingInterval	10000
AvoidLoopback	yes

Bridge
No description available.

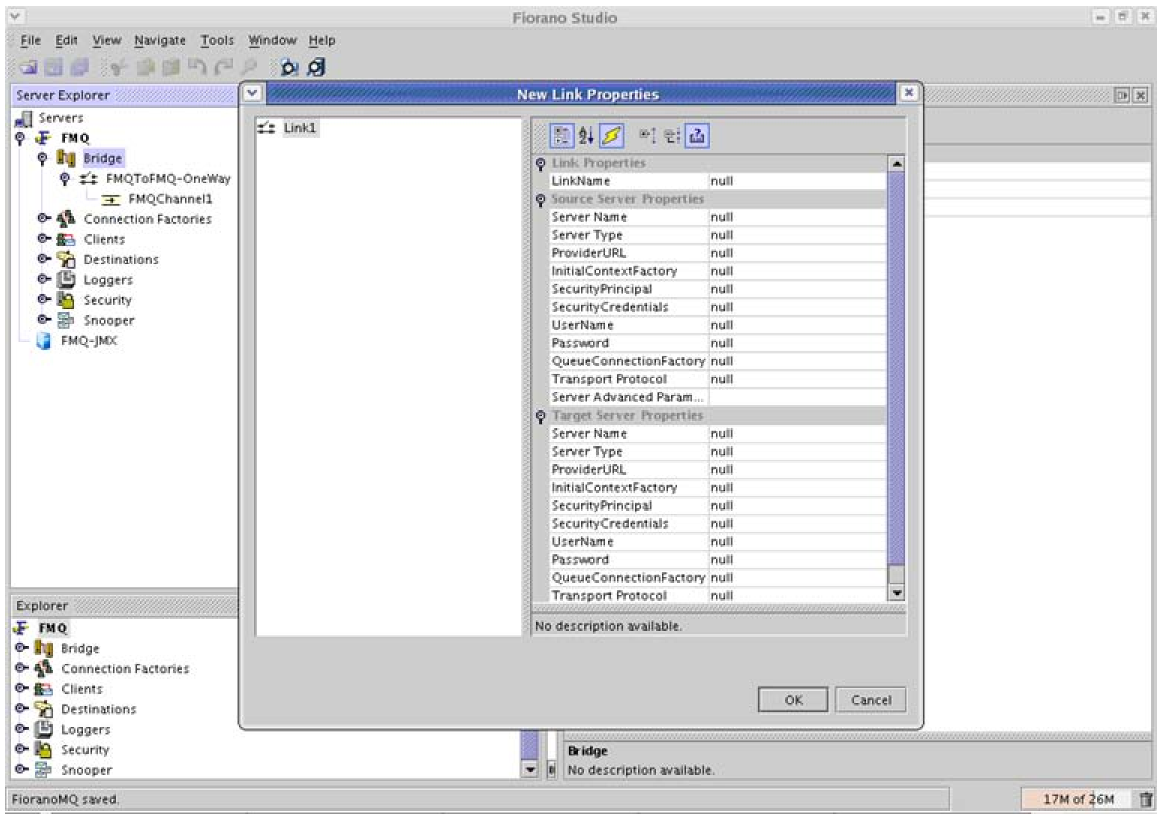
FioranoMQ saved.

3. In the **New Link Properties** dialog box, click the show expert properties icon (which is yellow in color) displayed at the top of the property sheet.



LinkName	null

All the required properties are shown on the property sheet.



Link Properties	
LinkName	null
Source Server Properties	
Server Name	null
Server Type	null
ProviderURL	null
InitialContextFactory	null
SecurityPrincipal	null
SecurityCredentials	null
UserName	null
Password	null
QueueConnectionFactory	null
Transport Protocol	null
Server Advanced Param...	
Target Server Properties	
Server Name	null
Server Type	null
ProviderURL	null
InitialContextFactory	null
SecurityPrincipal	null
SecurityCredentials	null
UserName	null
Password	null
QueueConnectionFactory	null
Transport Protocol	null

No description available.

Bridge
No description available.

- Add the appropriate values to the required properties. Server Advanced Params' can remain empty. Studio will display the new configurations. Detailed configurations are shown only if **show expert property** is enabled.

4. Right-click on the new link and perform the **add channel** operation.

5. Assign the appropriate valued through the input wizard. Properties are non editable. Once the configuration is added, a User is unable to edit parameter values. If the User needs to alter these values then the link or the channel should be deleted and replaced with a link and/or a channel that has the required parameters.

12.4 Configuring FioranoMQ Bridge for other Messaging Servers

The next section explains the steps needed to configure the FioranoMQ Bridge connecting the FioranoMQ with other messaging servers.

Note: Tested using the following versions of other vendors

- ActiveMQ 5.3
- Jboss 5.0.1GA
- MQSeries 7.1
- MSMQ version 3
- OpenMQ 4.4

12.4.1 To Configure the Bridge

Execute the following steps:

Rename the configuration .xml file as **Configs.xml**. FioranoMQ ships the following .xml files to configure the FioranoMQ Bridge with other messaging servers:

- Bridge(Jboss).xml
- Bridge(ActiveMQ).xml
- Bridge(OpenMQ).xml
- Bridge(ibmMQ).xml
- Bridge(Tibrv).xml
- Bridge(TibEms).xml
- Bridge(MSMQ).xml

These xml files are available in the **fmq\profiles\StandAloneBridge\conf** directory in the FioranoMQ installation package.

Note: Since FioranoMQ's 2008SP2 release, a text based configuration file called **Configs.cfg** has been added to the **fmq\profiles\StandAloneBridge\conf** folder. For more information on this, please refer to Chapter 2 of FioranoMQ Getting Started guide. This file has not yet been modified to automatically provide configurations when Fiorano's Bridge is used to connect to other vendors. When FioranoMQ server boots up, configuration parameters are picked up from the **Configs.cfg** file. It is mandatory that the **Configs.cfg** file be renamed so as to avoid undesirable results. Parameters will be picked up only from the Configs.xml file.

Configuration Steps

The StandAloneBridge profile comes up with a set of pre-configured Bridge links between the FioranoMQ server and other JMS vendors. The configuration steps below can be applied to any of the above .xml files for sending messages from one server to another.

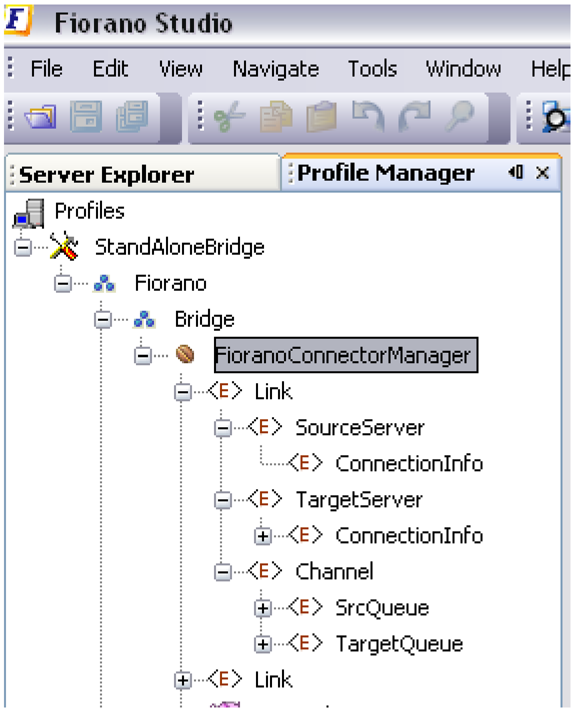
FioranoMQ StandAloneBridge can be used by JMS providers supported by FioranoMQ.

- The default Configs.xml file links two FioranoMQ servers transferring messages from one Queue to a different Queue. In order to use the pre-configured configuration file Bridge (SonicMQ).xml, please rename this file as Configs.xml.

Each configuration file links a Queue formed in the FioranoMQ server to a Queue formed in the corresponding non-FioranoMQ messaging server. [

Open the StandAloneBridge profile through the Fiorano Studio to configure the links or to check the configuration of the **FMQToSonicMQ (considering Bridge()SonicMQ.xml file)** links.

- Each link is configured with a 'Source Server' and a 'Target Server' along with connection information (connectionInfo as in the diagram) as well as a Channel, as shown in the figure below:



The 'SourceServer' component is configured as FioranoMQ and does not require additional parameters.

The 'TargetServer' component is configured for the 'SonicMQ' server with 5 additional (Name, Value) pair parameters. Please note that a new parameter can be created by right-clicking on 'ConnectionInfo' and selecting the 'Add Parameter'. By default NO Parameters are added to a new Link.

The Channel component should have SrcQueue (FioranoMQ) and the TargetQueue (SonicMQ).

Add the following jars to the fmq.conf file under '<java.classpath>' tag with their path set to:

- mfcontext.jar
- sonic_Client.jar

Please note that the vendor's client jars and libraries should be in the classpath that is defined at the time of running the FioranoMQ Bridge. Modify the fmq.conf file available in the **fmq\bin** directory of the FioranoMQ installation package.

Under the java.classpath tag there are different sets of jar files for each vendor. Uncomment the set of files corresponding to the vendor after modifying it to point to the right path.

- Start the FioranoMQ server and then start the **SonicMQ/JBoss/OpenMQ/ActiveMQ/IBM MQ/Tibrv** messaging servers.
- Test the Bridge by running the samples while taking care of the source and target servers.
- The Bridge can be configured either offline or online using the Admin Studio.

12.4.1.1 MSMQ Instructions

In the MSMQ Bridge, private Queues can be created only on the local computer. Queue name in the **Configs.xml** file should be **private$\QueueName**. Public Queues can be created only when the MSMQ installed computer is in the domain since public queues use Active Directory Services provided by the domain controller. Public queues can be created on local and as well as remote computers with Queue names as **QueueName** in the **Configs.xml** file.

12.4.1.2 OpenMQ Instructions

In the OpenMQ bridge, configure Object store and add the administrated objects (Connection Factory and Destination) which are to be used. In the default Setting of the **Bridge(OpenMQ).xml**, the connection factory used is- **MyConnection Factory** and the Destination used is **MyQueue**. therefore, prior to running the Bridge profile, add the two objects in the object store. Create this destination or any other destination to be Bridged on the **OpenMQ** broker. For more information on configuring OpenMQ refer to: http://docs.sun.com/app/docs/doc/819-7755/6n9m8u57v?a=view#aeoay.

Note: If the ForeignMQ version is supported on jdk1.6 or later versions, change the environmental variable (JAVA_HOME and JDK_HOME) accordingly. Windows system provides the new JAVA_HOME path in **fiorano_vars.bat**.

12.4.1.3 JBoss Messaging Instructions

JBoss Messaging supersedes JBoss MQ as the default Java Message Service (JMS) provider in JBoss Application Server (JBoss AS) 5. The jars to be added in the classpath while starting FioranoMQ Bridge are, therefore, slightly different for JBoss Messaging. These jars are:,

- $JBOSS_HOME/client/jboss-messaging-client.jar
- $JBOSS_HOME/client/jnp-client.jar
- $JBOSS_HOME/client/jboss-aop-client.jar
- $JBOSS_HOME/client/xmlsec.jar
- $JBOSS_HOME/client/jboss-serialization.jar
- $JBOSS_HOME/client/jboss-mdr.jar
- $JBOSS_HOME/client/jboss-logging-spi.jar
- $JBOSS_HOME/client/jboss-remoting.jar
- $JBOSS_HOME/client/trove.jar
- $JBOSS_HOME/client/javaassist.jar
- $JBOSS_HOME/client/concurrent.jar
- $JBOSS_HOME/client/log4j.jar

Note: JBOSS_HOME is the location where JBoss AS is extracted/installed.

The JBoss Messaging queue specified in the **Configs.xml** of the FioranoMQ Bridge profile must exist. This can be done by editing the file **$JBOSS_HOME/server/default/deploy/jms/jbossmq-destinations-service.xml** to include the queue with the appropriate properties. For example, to create testQueue with permission for Users to send messages on it, the following **Mbean** must be added:

```
  <mbean code="org.jboss.jms.server.destination.QueueService"
    name="jboss.messaging.destination:service=Queue,name=testQueue"
    xmbean-dd="xmdesc/Queue-xmbean.xml">
    <depends optional-attribute-
name="ServerPeer">jboss.messaging:service=ServerPeer</depends>


    <attribute name="SecurityConfig">
         <security>
            <role name="publisher" read="true" write="true" create="false"/>
         </security>
      </attribute>
      <depends>jboss.messaging:service=PostOffice</depends>
   </mbean>
```

Chapter 13: Dead Message Queue

This section of the document provides instructions on enabling DMQ and expired message notifications.

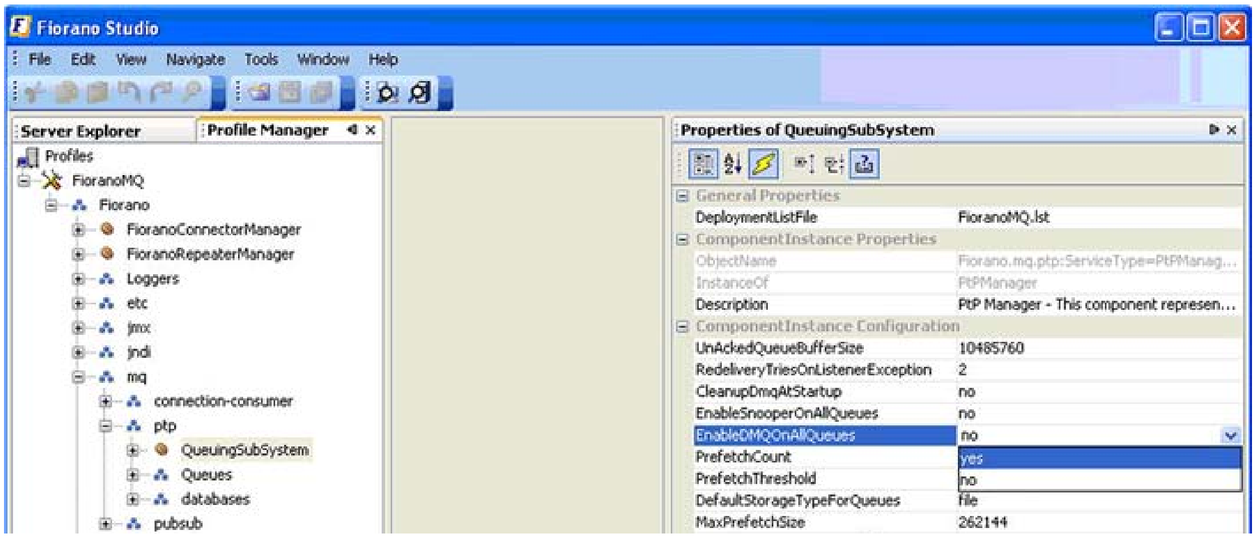
13.1 Editing Global/Default DMQ Configuration

- Launch **Fiorano Studio** and open the appropriate profile through the Profile Manager
- The tree displayed shows all the queues and topics created under the node **ptp -> queues** is displayed.

Note: The queues are shown in off-line mode only if the server has been started with the same profile at least once before.

- Navigate to Fiorano -> mq -> ptp -> QueingSubSystem.
- DMQ related properties that are editable through the properties panel of this node are listed in the table below:

S.No	Parameter	Description
1	DMQExpiryTime	The amount of time (in milliseconds) thatmessages live on DMQ.
2	CleanupDmqAtStartup	If set to **Yes**, all DMQ messages are deleted upon server startup.
3	EnableDMQOnAllQueues	Boolean value of DMQ status (**yes/no**) for all queues that have the DMQEnabled property configured to "Default".



General Properties	
DeploymentListFile	FioranoMQ.lst
ComponentInstance Properties	
ObjectName	Fiorano.mq.ptp:ServiceType=PtPManag...
InstanceOf	PtPManager
Description	PtP Manager - This component represen...
ComponentInstance Configuration	
UnAckedQueueBufferSize	10485760
RedeliveryTriesOnListenerException	2
CleanupDmqAtStartup	no
EnableSnooperOnAllQueues	no
EnableDMQOnAllQueues	no
PrefetchCount	yes
PrefetchThreshold	no
DefaultStorageTypeForQueues	file
MaxPrefetchSize	262144

13.2 Disabling DMQ on the message level

An application can disable the DMQ function on a 'per message' basis by invoking the API, below, on a message prior to sending the message.

```
public void setStoreWhenDead(boolean value)

        throws FioranoException
```

Note: Since this is a proprietary API, using it requires that the application include the import statements below:

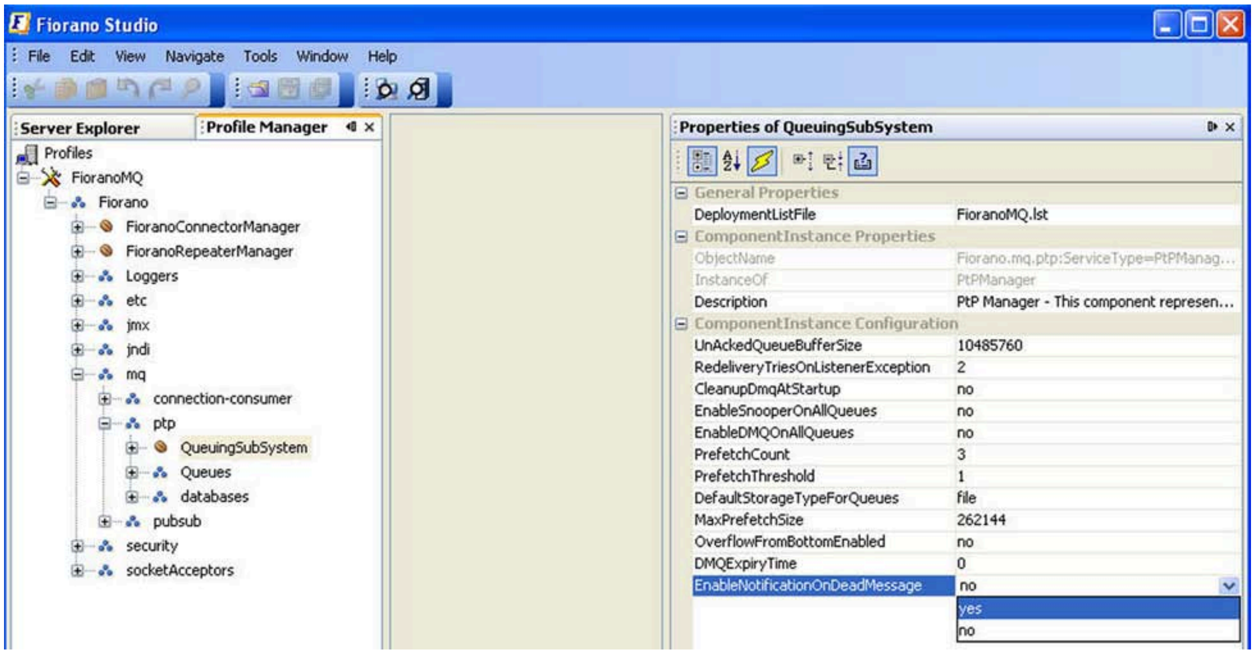
```
import fiorano.jms.services.msg.def.FioranoMessage;

import fiorano.jms.common.FioranoException;
```

This API is effective only when the DMQ is enabled. If DMQ is not enabled, using this API with the boolean value set to **true** will not have any effect.

13.3 Enabling Notifications for Expired Messages

- Launch **Fiorano Studio** and open the appropriate profile usingthe **Profile Manager** tab.
- The tree displayed shows all the queues and topics created under the node **ptp -> queues & pubsub -> Topics**, as shown in the figure below.

  Note: The queues are shown in the off-line mode only if the server has been started with the same profile at least once before.
- Navigate to Fiorano -> mq -> ptp -> QueingSubSystem, as shown in the figure below, and edit the property EnableNotificationsOnDeadMessage.



General Properties	
DeploymentListFile	FioranoMQ.lst
ComponentInstance Properties	
ObjectName	Fiorano.mq.ptp:ServiceType=PtPManag...
InstanceOf	PtPManager
Description	PtP Manager - This component represen...
ComponentInstance Configuration	
UnAckedQueueBufferSize	10485760
RedeliveryTriesOnListenerException	2
CleanupDmqAtStartup	no
EnableSnooperOnAllQueues	no
EnableDMQOnAllQueues	no
PrefetchCount	3
PrefetchThreshold	1
DefaultStorageTypeForQueues	file
MaxPrefetchSize	262144
OverflowFromBottomEnabled	no
DMQExpiryTime	0
EnableNotificationOnDeadMessage	no

13.4 Subscribing to Notifications for Expired Messages

When a message expires, the server, if configured (as above), publishes a notification in the form of a JMS Text message on a system topic named ADMINISTRATOR_TOPIC. In order to receive this notification, a simple Subscriber can be created on this topic as per the JMS Semantics.

13.5 Disabling Expiry Notifications on a message level

An application can disable the notification function on a ‘per message’ basis by invoking the following API on a message before sending the same.

public void setNotifyWhenDead(boolean value)

throws FioranoException

Note:

Since this is a proprietary API, using it requires that the application include the import statements below:

import fiorano.jms.services.msg.def.FioranoMessage;

import fiorano.jms.common.FioranoException;

This API is effective only when the DMQ and the ‘notification’ are enabled on the queue.

Chapter 14: Named Configuration

Named configurations are predefined configurations that are assigned a name and stored for later reuse. These named configurations ease the process of Event Process Orchestration and Change Management within the Fiorano Event Processes. For example, if a particular connection configuration for SMTP or JMS components is reused in multiple Event Processes, each such Service Instance will have its own copy of the configuration. If a change in configuration is required at a later point of time, then all such Service Instances have to be reconfigured. Using named configuration support, configurations can be predefined and the name of the predefined configuration can be linked/transferred to all Service Instances. Since the actual configuration has only one location but is referred to by multiple Service Instances, making changes to the named configuration will affect all Service Instances automatically (without the need to reconfigure the Service Instances again). The route messaging/selector/transformation configuration, port messaging/workflow configuration and runtime arguments/connection factory properties of service instances follow the same principle stated above.

Terminology

Below is a glossary of terms used in this Chapter.

Term Used	Meaning
Named Configuration / Named Object	A name-value pair that stores the configuration given against the name specified
Registry	A location within which all named configurations are stored
Artifact	An artifact can refer to any entity in an Event Process that requires configuration. For example, service instances, routes, ports are some artifacts.

Chapter 15: Hierarchical Topics

15.1 Creating a Hierarchical Topic

15.1.1 Admin API

The User can create Hierarchical Topics with the help of AdminAPIs. The code snippet below demonstrates the process of the creation of topics in Hierarchical name spaces.

Note:

1. Queue, Topic and ConnectionFactory names cannot have any of these characters * # \\ / : | ? " < >
2. To create a Topic, it is necessary that the parent to the topic exists:

To create a Topic, it is necessary that the parent to the topic exists:

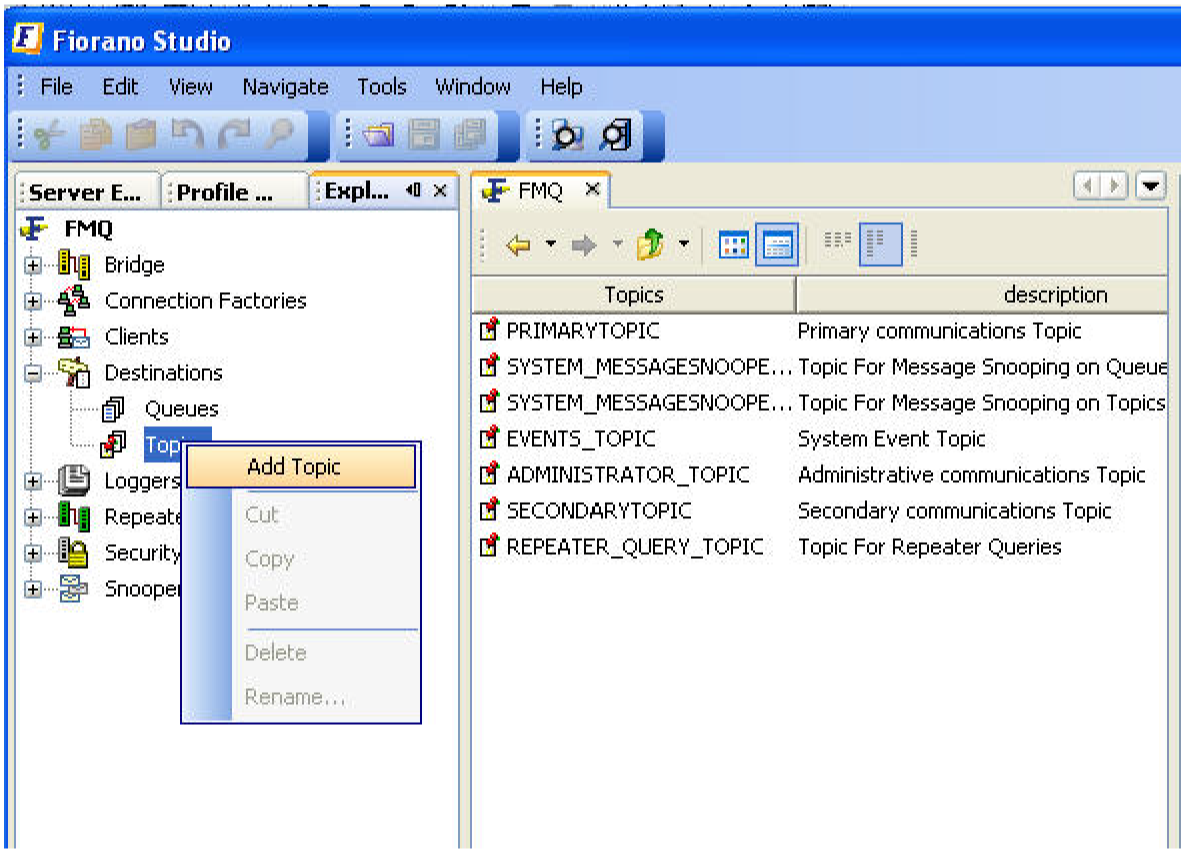
```
public void testHierarchicalTopics ()

{

// create the initial context and connect to FioranoMQ 9

Hashtable env = new Hashtable ();

env.put (Context.SECURITY_PRINCIPAL, "anonymous");

env.put (Context.SECURITY_CREDENTIALS, "anonymous");

env.put (Context.PROVIDER_URL, "http://localhost:1856");

env.put
Context.INITIAL_CONTEXT_FACTORY,"fiorano.jms.runtime.naming.FioranoInitialContextFactor
y");

InitialContext ic = new InitialContext (env);



System.out.println ("Created InitialContext:: " + ic);

MQAdminConnectionFactory acf =

(MQAdminConnectionFactory) ic.lookup ("primaryACF");

MQAdminConnection ac = acf.createMQAdminConnection ("admin", "passwd");

System.out.println ("Created Admin Connection...");
```

```
MQAdminService adminService = ac.getMQAdminService ();

System.out.println ("Received handle to Admin services:: " + adminService);


// Create a topic named primaryTopic.subTopic1

String topicName = "primarytopic.subtopic1";

TopicMetaData tMetaData = new TopicMetaData();

tMetaData.setName (topicName);

adminService.createTopic (tMetaData);

Topic topic = (Topic) ic.lookup (topicName);

System.out.println ("Looked up the Hierarchical Topic ", topic.getName());


//create a topic named primarytopic.subtopic2

String topicName = "primarytopic.subtopic2";

TopicMetaData tMetaData = new TopicMetaData ();

tMetaData.setName (topicName);

adminService.createTopic (tMetaData);

topic = (Topic) ic.lookup (topicName);

System.out.println ("Looked up the Hierarchical Topic ", topic.getName());

}
```

15.1.2 Studio

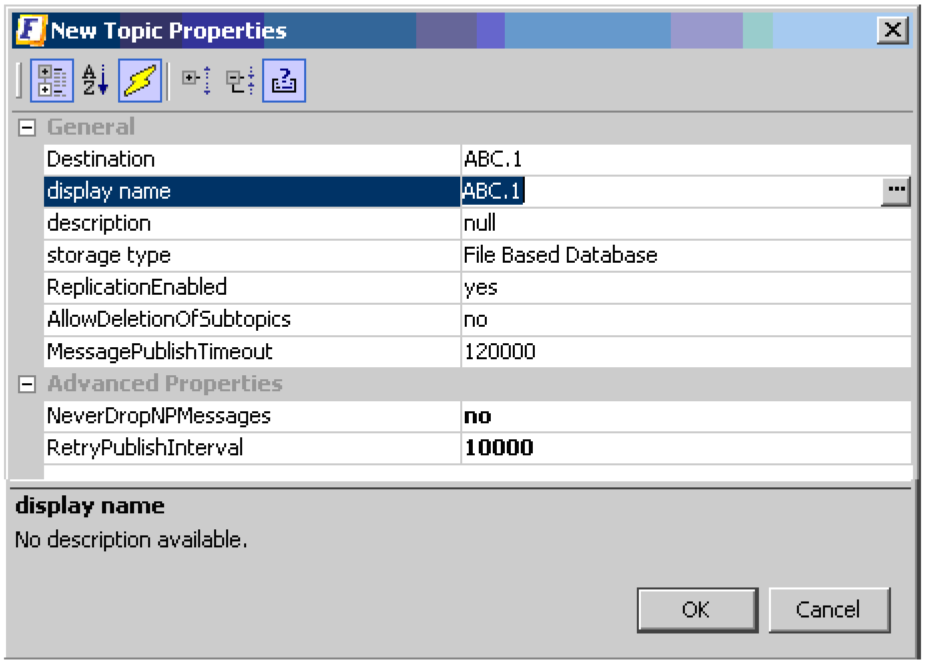
The User can create Hierarchical Topics with admin Studio. To create hierarchical topics with studio, follow the steps below:

3. Launch **Fiorano Studio** and connect Studio to the FioranoMQ Server.
4. To create a topic, navigate to the **Topics** sub node under the **Destinations** node of the tree
5. Right-click on the **Topics** node and select **Add Topic**



Topics	description
PRIMARYTOPIC	Primary communications Topic
SYSTEM_MESSAGESNOOPE...	Topic For Message Snooping on Queue
SYSTEM_MESSAGESNOOPE...	Topic For Message Snooping on Topics
EVENTS_TOPIC	System Event Topic
ADMINISTRATOR_TOPIC	Administrative communications Topic
SECONDARYTOPIC	Secondary communications Topic
REPEATER_QUERY_TOPIC	Topic For Repeater Queries

The window below is displayed:

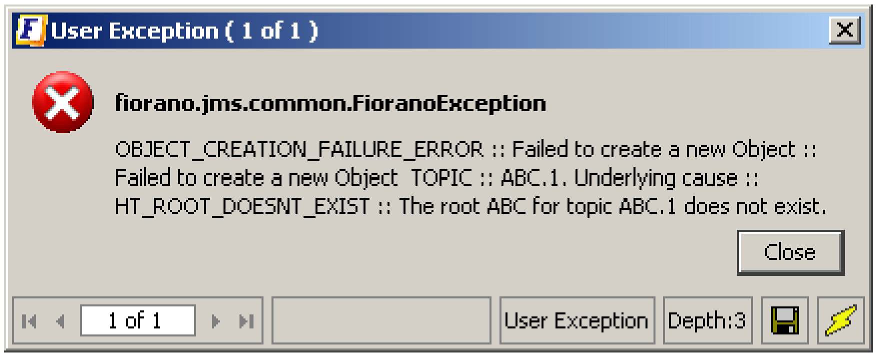


General	
Destination	ABC.1
display name	ABC.1
description	null
storage type	File Based Database
ReplicationEnabled	yes
AllowDeletionOfSubtopics	no
MessagePublishTimeout	120000
Advanced Properties	
NeverDropNPMessages	**no**
RetryPublishInterval	**10000**

display name
No description available.

6. Enter the Destination parameter and other relevant parameters necessary for the creation of the topic. In this example, the Topic name is ABC.1

Note: For the creation of topicName ABC.1, a parent topic should exist. In the absence of ABC, the following error is displayed on the console:



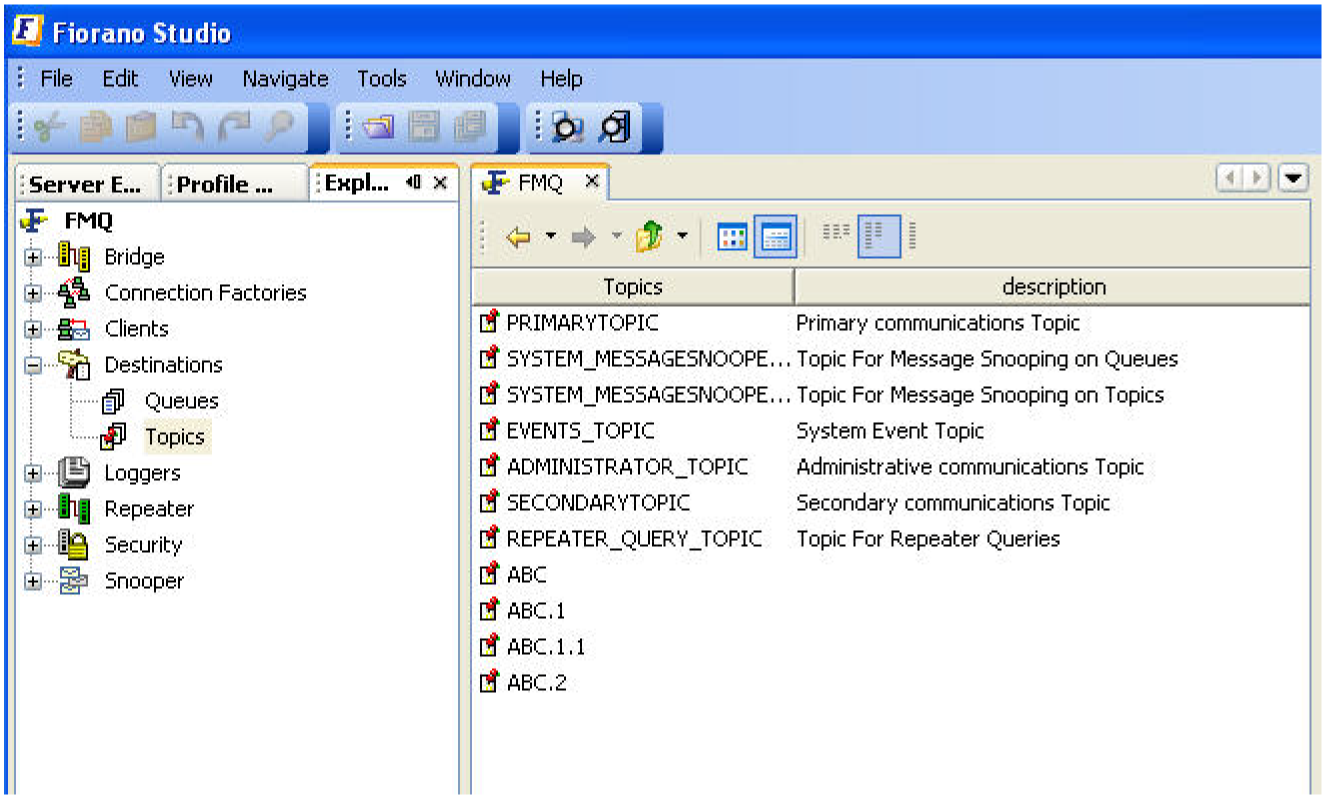
User Exception (1 of 1)

fiorano.jms.common.FioranoException

OBJECT_CREATION_FAILURE_ERROR :: Failed to create a new Object :: Failed to create a new Object TOPIC :: ABC.1. Underlying cause :: HT_ROOT_DOESNT_EXIST :: The root ABC for topic ABC.1 does not exist.

7. In the same manner, the user can create the topics hierarchy

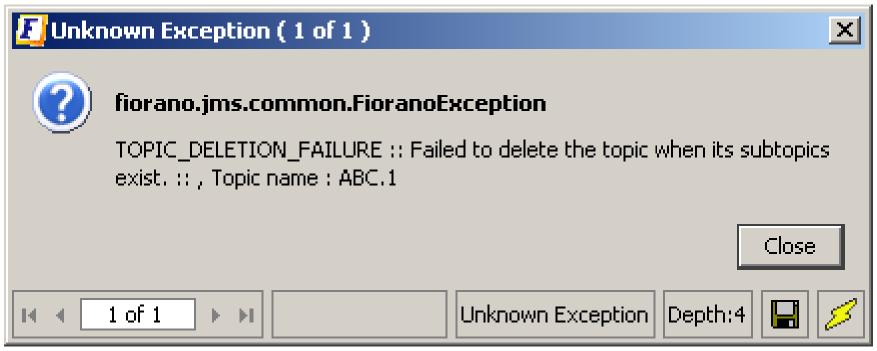
Once a hierarchy is created, all the topics are displayed in the **Topic List**.



Topics	description
PRIMARYTOPIC	Primary communications Topic
SYSTEM_MESSAGESNOOPE...	Topic For Message Snooping on Queues
SYSTEM_MESSAGESNOOPE...	Topic For Message Snooping on Topics
EVENTS_TOPIC	System Event Topic
ADMINISTRATOR_TOPIC	Administrative communications Topic
SECONDARYTOPIC	Secondary communications Topic
REPEATER_QUERY_TOPIC	Topic For Repeater Queries
ABC	
ABC.1	
ABC.1.1	
ABC.2	

15.2 Deleting a Hierarchical Topic

Deletion of a topic/subtopic from the hierarchical name space depends on the value of the parameter **AllowDeletionOfSubTopics**, which can be configured through Fiorano Studio. If this value is set to **true**, then deletion of a topic/subtopic deletes all the children of this topic/subtopic. However, if it is set to **false**, the following exception is raised, indicating that the User needs to first delete the children of the topic/subtopic before deleting the topic itself.

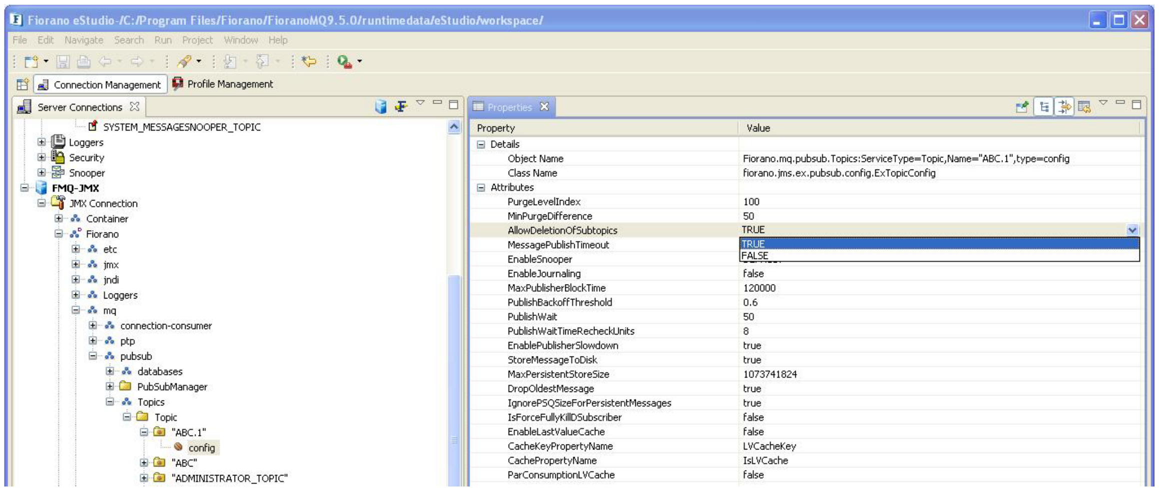


fiorano.jms.common.FioranoException

TOPIC_DELETION_FAILURE :: Failed to delete the topic when its subtopics exist. :: , Topic name : ABC.1

By default, this variable is set to **false**. Follow the steps given below to delete a hierarchical topic\subtopic.

1. Start Fiorano **eStudio/Studio** and login to **FMQ-JMX**
2. Select the Topic config from **JMX Connection-->Fiorano-->mq-->PubSub-->Topic-->${TopicName}-->config**
3. Enable the value of the parameter named **AllowDeletionOfSubtopics** (GeneralPropeties) by setting it up to **TRUE** from the drop-down list.



Property	Value
Details	
Object Name	Fiorano.mq.pubsub.Topics:ServiceType=Topic,Name="ABC.1",type=config
Class Name	fiorano.jms.ex.pubsub.config.ExTopicConfig
Attributes	
PurgeLevelIndex	100
MinPurgeDifference	50
AllowDeletionOfSubtopics	TRUE
MessagePublishTimeout	
EnableSnooper	
EnableJournaling	false
MaxPublisherBlockTime	120000
PublishBackoffThreshold	0.6
PublishWait	50
PublishWaitTimeRecheckUnits	8
EnablePublisherSlowdown	true
StoreMessageToDisk	true
MaxPersistentStoreSize	1073741824
DropOldestMessage	true
IgnorePSQSizeForPersistentMessages	true
IsForceFullyKillDSubscriber	false
EnableLastValueCache	false
CacheKeyPropertyName	LVCacheKey
CachePropertyName	IsLVCache
ParConsumptionLVCache	false

15.3 Setting up Security on a HT

FioranoMQ supports ACL settings for Hierarchical topics. An ACL can be set for any topic, irrespective of the level at which the topic exists. These ACLs are checked at the time of creating a publisher as well as at the time of creating a subscriber. While creating a subscriber on multiple topics (a topic that involves a template character in its name), the ACLs of all subtopics are also checked. In addition, the subscriber is modified so that it does not receive messages from subtopics that have a negative permission set for that particular User.

15.4 Looking Up a HT

A client application can look up a topic in the FioranoMQ Server using either JNDI APIs or a bound object of type FioranoInitialContext. Criteria for looking up Hierarchical topics are given below:

15.4.1 Wild Characters * or

The topic being looked up contains a wildcard character '*' or '#' with any number of delimiters. A delimiter (.) can not be the last character of a topic name that is looked up.

The look up call succeeds only if the root topic has been created by the administrator at an earlier stage. If the topic being looked up contains a '*' or '#' then this call is successful only when there is at least one topic existing in the server whose name matches the criterion. For example: If the user tries to look up "primarytopic.a.*" or "primarytopic.a.#" then the look up call is successful only if "primarytopic.a" exists.

15.5 Prerequisites

In FioranoMQ Hierarchical Topics support, the user can create a subtopic in a hierarchy on a server that is running. For dynamic topic creation support, Events should be turned when the server is running.

15.5.1 Events to be turned on for dynamic topic creation support

If a topic is created on a running server instance and its name matches any subscription expressions (if they exist) then this topic becomes a member of the maintained hierarchy for subscription on Hierarchical topics.

Example

Subscription expression: ABC.*

Topics existing on the system: ABC, ABC.1, ABC.2, ABC.1.1

A subscriber looks up a topic with the expression ABC.* and receives messages from matching topics. At runtime a new topic named 'ABC.3' (which does not exist in the created hierarchy) is created. This new topic becomes part of the hierarchy and published messages on ABC.3 are also received by the Subscriber created on ABC.*.

15.6 HT Limitations

Topic names cannot contain a wildcard character. For subscription expressions, no other template character (‘*’ ‘or’ # with any number of dots “.”) is used. Usage of any other template character throws an exception in the look up call. All attempts to delete hierarchical topics are successful whether or not the topic has active publishers/subscribers present. It is, therefore, important to be careful so as not to delete hierarchical topics while they contain any active publishers or active subscribers.

A publisher cannot publish on multiple topics. A publisher has to specify the complete name of the hierarchical topic on which it wants to publish data. Creation of a publisher on a topic, which contains an asterisk ‘*’, throws an exception. Similarly, an exception is thrown if a publisher tries to publish on a topic which contains an asterisk ‘*’. If a subscriber subscribes on hierarchical topics with a subscription expression and, while receiving messages, the administrator changes the ACL of one of the children of the hierarchy, then the subscriber will not be affected by this change. However, all new subscribers with a subscription expression will be affected.

There is a performance degradation associated with hierarchical topics. Users are, therefore, advised not to use hierarchical topics for applications where performance is a major requirement.

1. A Hierarchical Topic is not supported by the following flags:
    - UseOptimizedTCPReceive = true
    - This implies that the Hierarchical Topic is not supported by the 7.2 runtime Layer.
2. Un-subscription for Hierarchical topics does not work. The User has to unsubscribe the subscribers created on behalf of hierarchical support manually, through **Studio**.

Chapter 16: Snooper

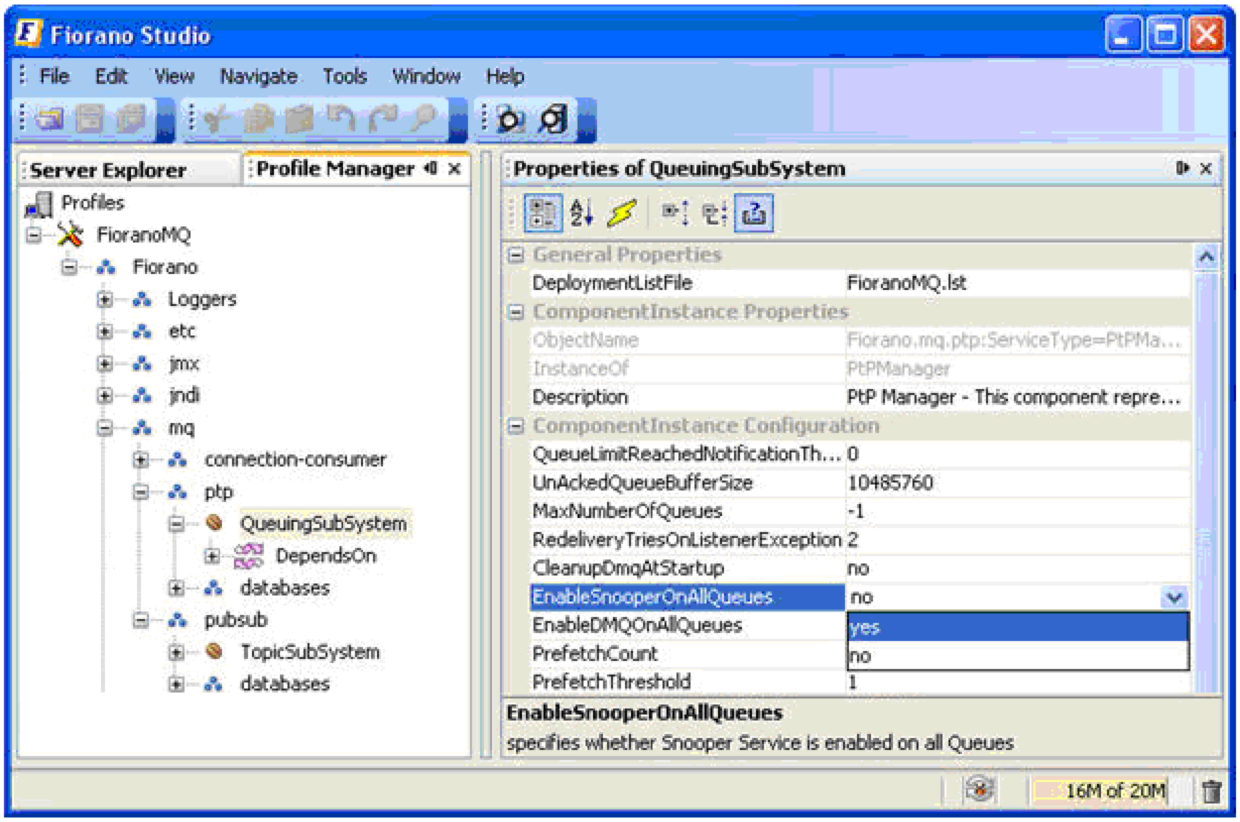
This section of the document provides detailed instructions regarding the Snooper function in FioranoMQ.

16.1 Editing the Snooper Configuration on a Destination

16.1.1 Editing the snooper Configuration on a Destination in the Offline Mode

For information about configuring profiles through a text based file, refer to FioranoMQ Getting Started.

1. Launch Fiorano **Studio** and open the required profile for offline editing through the **Profile Manager** tab. The tree displayed shows all queues and topics created under the node **ptp** and **pubsub** as shown in the figure below.
2. Navigate to **QueuingSubSystem** or **TopicSubSystem** and edit the property **EnableSnooperOnAllQueues** or **EnableSnooperOnAllTopics** entering the value required.



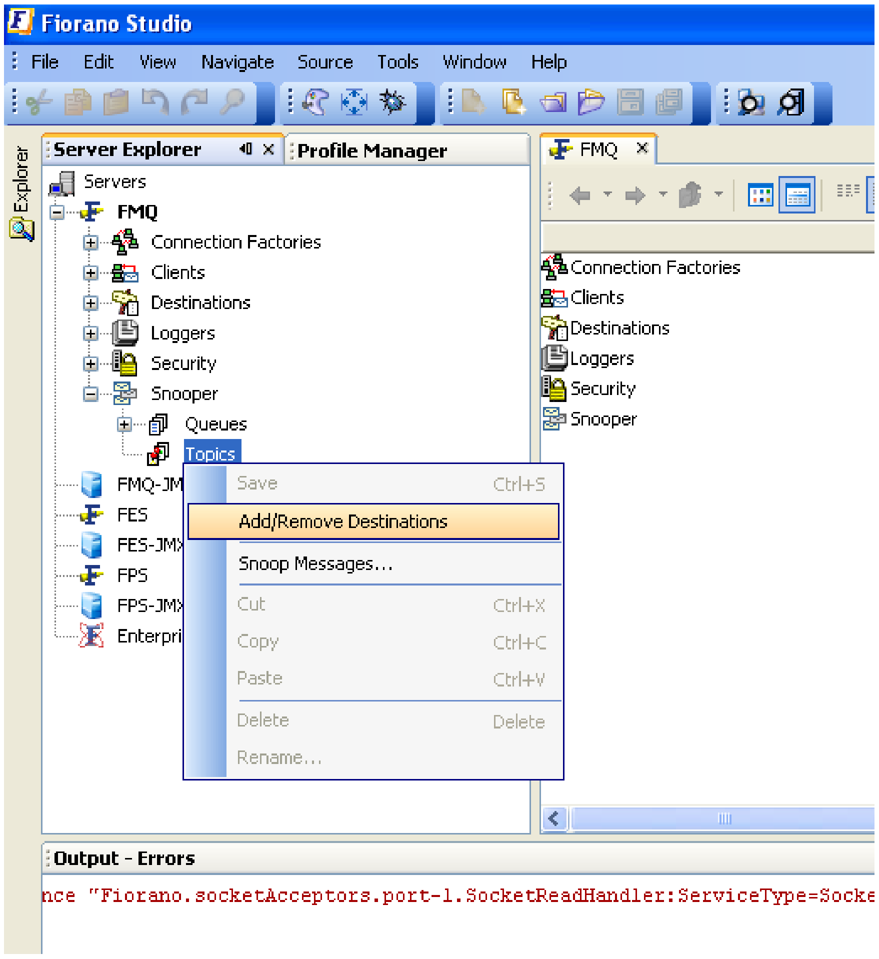
General Properties	
DeploymentListFile	FioranoMQ.lst
ComponentInstance Properties	
ObjectName	Fiorano.mq.ptp:ServiceType=PtPMa...
InstanceOf	PtPManager
Description	PtP Manager - This component repre...
ComponentInstance Configuration	
QueueLimitReachedNotificationTh...	0
UnAckedQueueBufferSize	10485760
MaxNumberOfQueues	-1
RedeliveryTriesOnListenerException	2
CleanupDmqAtStartup	no
EnableSnooperOnAllQueues	no
EnableDMQOnAllQueues	yes
PrefetchCount	no
PrefetchThreshold	1

EnableSnooperOnAllQueues
specifies whether Snooper Service is enabled on all Queues

3. **Right-click** the profile and select **Save** from the pop-up menu.

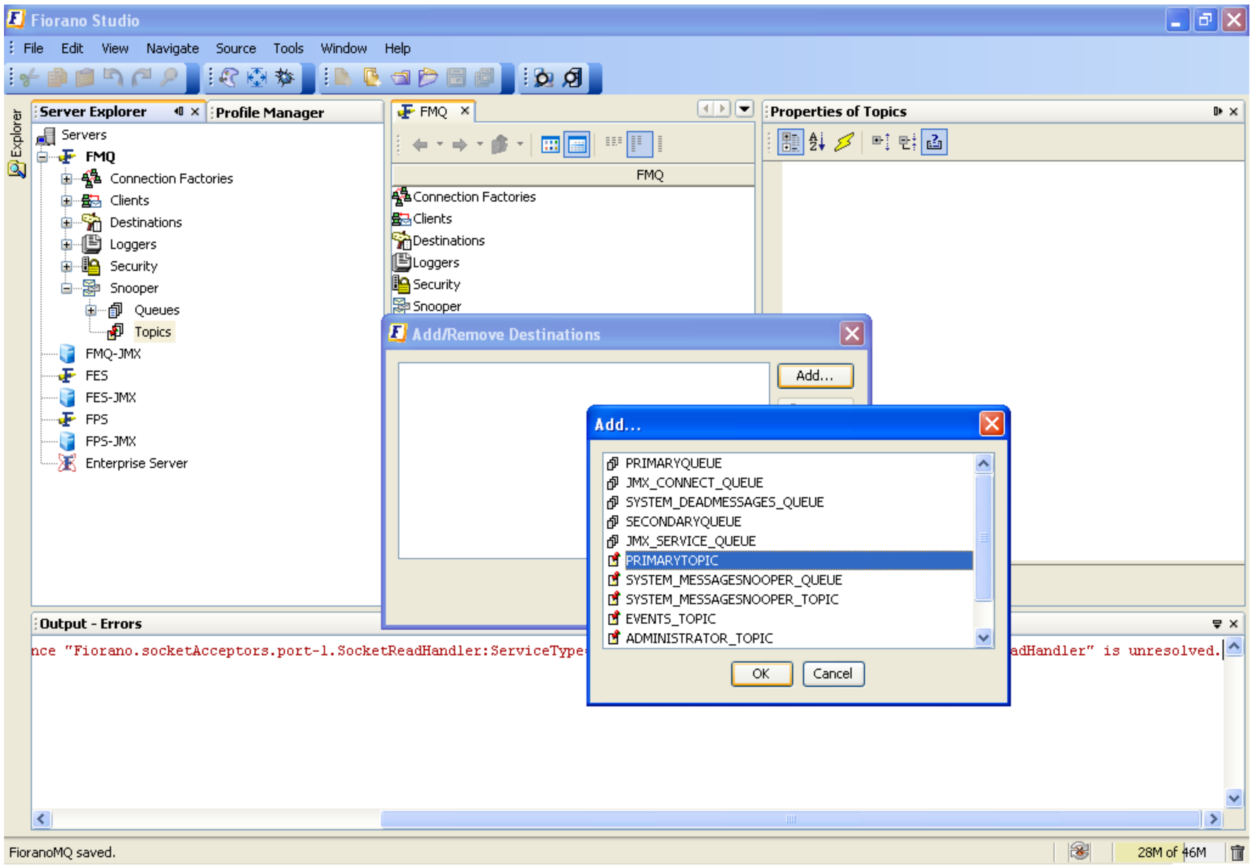
16.1.2 Editing the snooper Configuration on a Destination in the Online Mode

1. Launch the **Admin Studio** and connect to the appropriate Server through the **Server Explorer** pane.
2. Select the **Snooper** node in the tree, right-click and select **Add /Remove Destinations** from the pop-up menu. The **Add/Remove** dialog box is displayed showing all the destinations for which Snooper is currently configured.



```
nce "Fiorano.socketAcceptors.port-1.SocketReadHandler:ServiceType=Socke
```

3. Click on the **Add** button and another dialog box **Add...** is displayed showing the remaining destinations for which snooper is displayed.
4. Select the desired destination(s) (multiple selections are allowed by pressing the **Ctrl** key on the keyboard) and then click the **OK** button.



Add...

PRIMARYQUEUE
JMX_CONNECT_QUEUE
SYSTEM_DEADMESSAGES_QUEUE
SECONDARYQUEUE
JMX_SERVICE_QUEUE
PRIMARYTOPIC
SYSTEM_MESSAGESNOOPER_QUEUE
SYSTEM_MESSAGESNOOPER_TOPIC
EVENTS_TOPIC
ADMINISTRATOR_TOPIC

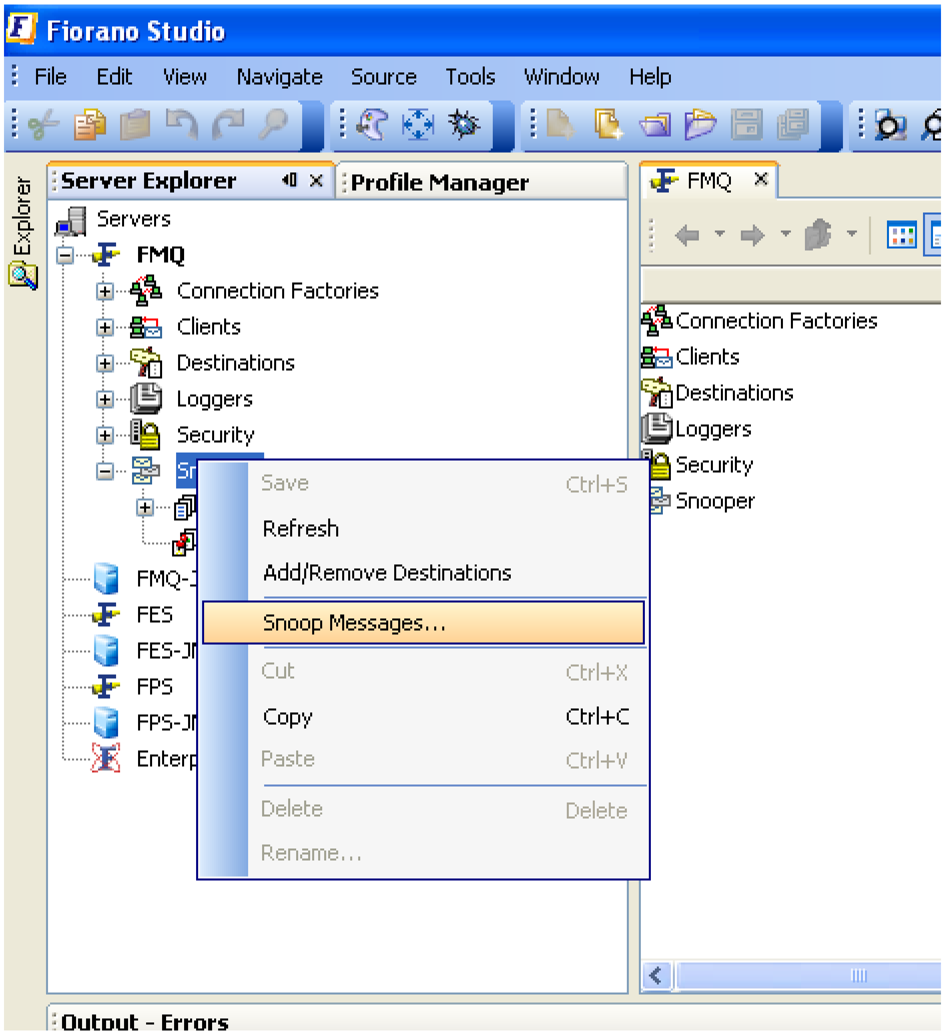
```
nce "Fiorano.socketAcceptors.port-1.SocketReadHandler:ServiceType
adHandler" is unresolved.
```

5. Right-click on the **FMQ** node and select **Save Configurations** from the pop-up menu.

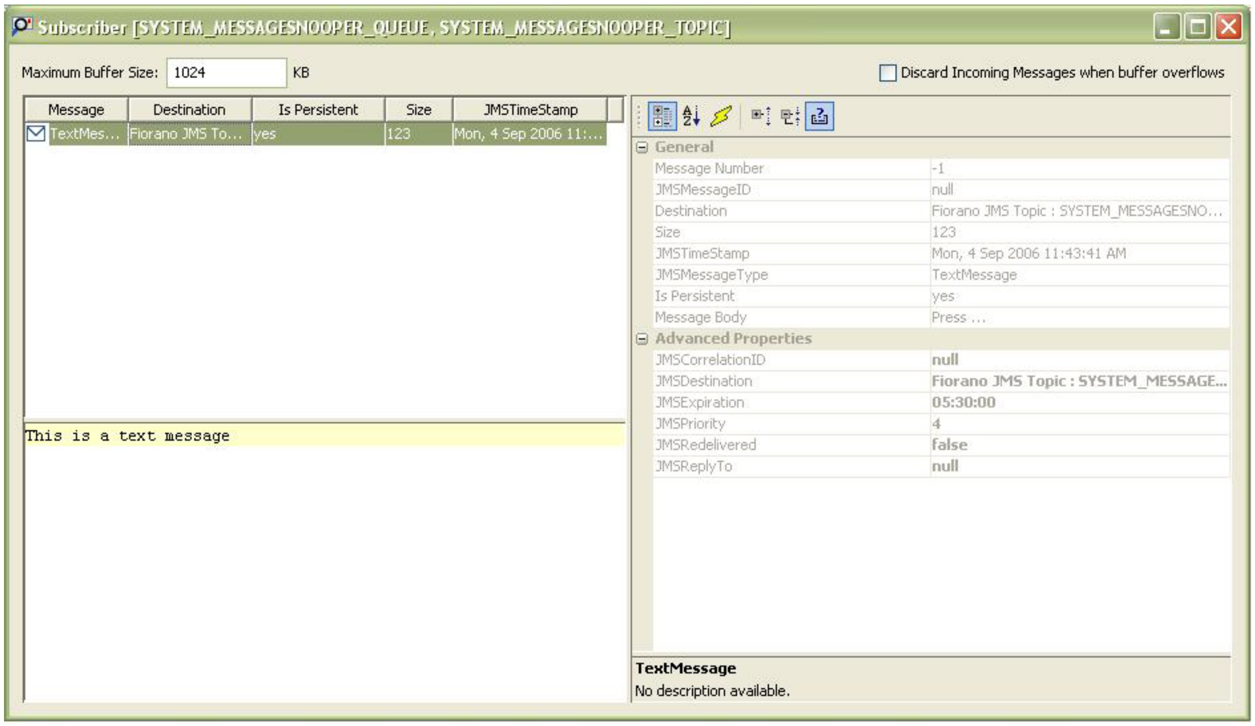
16.2 Viewing Snooped Messages

16.2.1 Studio

1. Launch **Admin Studio** and connect to a FioranoMQ Server through the **Server Explorer** Pane.
2. Select the **Snooper** node in the tree and right-click and select **Snoop Messages** from the pop-up menu as shown in the figure below,.



3. This opens a new window displaying Snooped messages.



Subscriber [SYSTEM_MESSAGESNOOPER_QUEUE, SYSTEM_MESSAGESNOOPER_TOPIC]

Maximum Buffer Size: 1024 KB

Discard Incoming Messages when buffer overflows

Message	Destination	Is Persistent	Size	JMSTimeStamp
TextMes...	Fiorano JMS To...	yes	123	Mon, 4 Sep 2006 11:...

This is a text message

General	
Message Number	-1
JMSMessageID	null
Destination	Fiorano JMS Topic : SYSTEM_MESSAGESNO...
Size	123
JMSTimeStamp	Mon, 4 Sep 2006 11:43:41 AM
JMSMessageType	TextMessage
Is Persistent	yes
Message Body	Press ...
Advanced Properties	
JMSCorrelationID	null
JMSDestination	Fiorano JMS Topic : SYSTEM_MESSAGE...
JMSExpiration	05:30:00
JMSPriority	4
JMSRedelivered	false
JMSReplyTo	null

TextMessage
No description available.

4. Selecting a message in this window displays all the properties as well as the body of the incoming message.

Note:

- To verify this setup, right-click on a queue (that has Snooper enabled) and select **produce a message**. This message should be visible in the Snooper window described above.
- The parameter **MaximumBufferSize** refers to the maximum amount of data that Studio holds for viewing. If the total data being stored exceeds this value, it clears the existing messages (or clears new incoming messages if the checkbox **Discard Incoming Messages if buffer full** is checked).

16.2.2 Programmatically

If an application Snoops messages on a destination, it needs to create Subscriber(s) on the System Snooper Topic, which is:

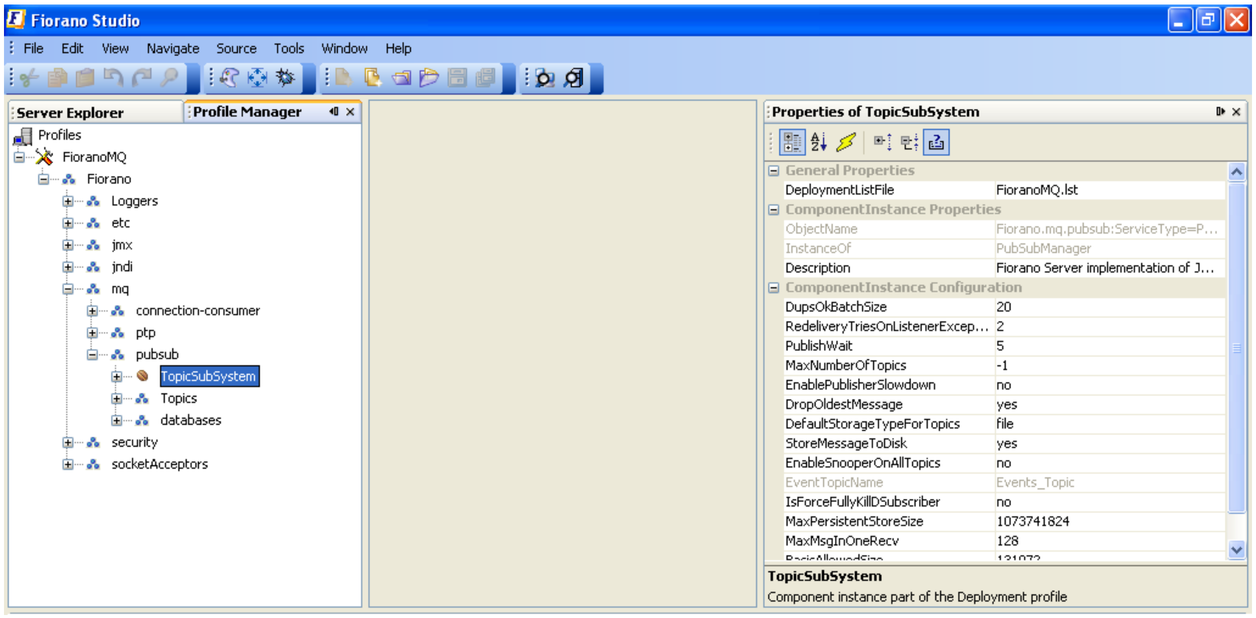
SYSTEM_MESSAGE_SNOOPER_QUEUE

The subscriber gets all messages published on a topic or a queue that has Snooper configured on it.

16.3 Editing the Default (Global) Configuration

16.3.1 Editing the Default Configuration in the Offline Mode

1. Launch **Fiorano Studio** and open the **Profile Manager**. Right-click on the **Profiles** node and select **Open Profile** from the pop-up menu. Select the required profile and click on the **Open** button.
2. In the component tree select the bean under the pubsub node as shown in the figure below.
3. Edit the parameter **EnableSnooperOnAllTopics** by entering the value required in the **Properties Panel**.



Properties of TopicSubSystem

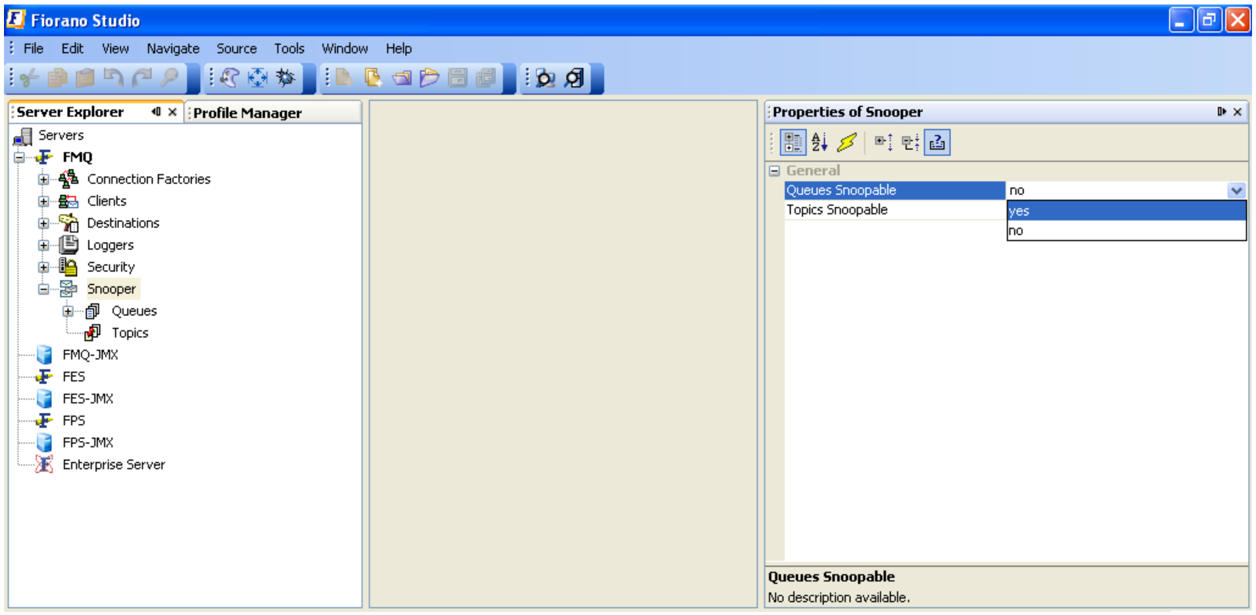
General Properties	
DeploymentListFile	FioranoMQ.lst
ComponentInstance Properties	
ObjectName	Fiorano.mq.pubsub:ServiceType=P...
InstanceOf	PubSubManager
Description	Fiorano Server implementation of J...
ComponentInstance Configuration	
DupsOkBatchSize	20
RedeliveryTriesOnListenerExcep...	2
PublishWait	5
MaxNumberOfTopics	-1
EnablePublisherSlowdown	no
DropOldestMessage	yes
DefaultStorageTypeForTopics	file
StoreMessageToDisk	yes
EnableSnooperOnAllTopics	no
EventTopicName	Events_Topic
IsForceFullyKillDSubscriber	no
MaxPersistentStoreSize	1073741824
MaxMsgInOneRecv	128

TopicSubSystem
Component instance part of the Deployment profile

4. In the tree shown in the figure above select the bean corresponding to **ptp**.
5. Modify the parameter **EnableSnooperOnAllQueues** as required in the **Properties Panel**.
6. **Right-click** the FioranoMQ node and select **Save** from the pop-up menu.

16.3.2 Online Mode

1. Launch the **Admin Studio** and connect to a running FioranoMQ Server through the **Server Explorer** pane.
2. From the tree shown in the figure below, navigate to the Snooper node and click on it.
3. Modify the values of the parameters **Queues Snoopable** and **Topics Snoopable** as shown in the figure below.



Queues Snoopable	no
Topics Snoopable	yes / no

Queues Snoopable
No description available.

16.4 Snooping Related Admin APIs

Almost all Snooper configuration settings can be accessed or edited through a program using Admin APIs. For an application to access this function, it must create an admin connection and obtain a reference for the **MQSnooperService**. The steps in establishing this connection are given below:

```
// Lookup Admin Connection Factory

AdminConnectionFactory acf =

              (AdminConnectionFactory) ic.lookup (“primaryACF”);



// Create Connection

MQAdminConnection ac = acf.createAdminConnection (“admin”, “password”);



// Get MQ Snooper Service

MQSnooperService snooperService = ac.getSnooperService ();

// do the desired operations through snooper Service
```

Details of APIs available in the Snooper Service can be found in Java Docs supplied along with the FioranoMQ installer.

Chapter 17: Message Journaling

Message Journaling is a FioranoMQ 2009 feature that allows an application to replicate the messages arriving at a destination to a destination of a similar 'type'. For example, if messages are sent to a Queue named 'PRIMARYQUEUE' and 'MessageJournaling' is enabled on this queue, then all the incoming messages are replicated and then sent to a journaling destination, say 'JOURNAL_PRIMARYQUEUE', which is also a Queue.

17.1 Using Message Journaling Feature

Message Journaling essentially means to copy a message from one destination to another destination of similar 'types'. So, when Journaling is enabled, a console-based application can be written in order to consume messages from the original destination as well as from the journaling destination. This application can programmatically receive the original produced message and inspect this message in any manner required.

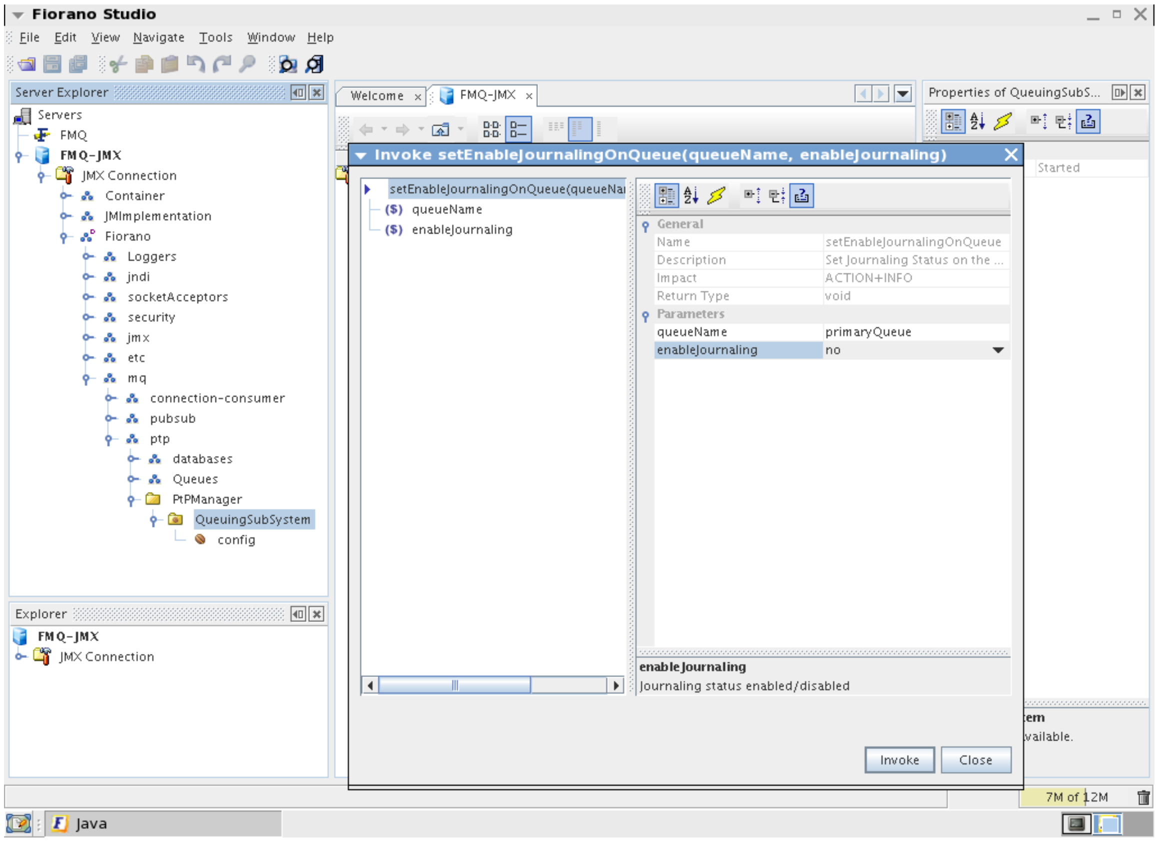
17.2 Configuring Message Journaling

17.2.1 Enabling Journaling flag

In order to replicate messages on a destination, the Journaling function on that destination has to be turned ON. This can be done through **Studio** in **online** mode or from the **Fiorano Web Management Tool**. Given below is the procedure that can be followed to enable journaling on Queues. Similar procedures can be followed for Topics also.

17.2.1.1 Online Mode

1. Launch **Admin Studio** and connect to a FioranoMQ 9 Server that is running through the **Server Explorer** pane.
2. Navigate to **QueueingSubSystem** node and right-click on it. From the list of APIs displayed, select **setEnableJournalingOnQueue**(queueName, enableJournaling).
3. Modify the values of the parameters **queueName** and **enablejournaling** as shown in the figure below.



General	
Name	setEnableJournalingOnQueue
Description	Set Journaling Status on the ...
Impact	ACTION+INFO
Return Type	void
Parameters	
queueName	primaryQueue
enableJournaling	no

enableJournaling
Journaling status enabled/disabled

4. Click the **Close** button.

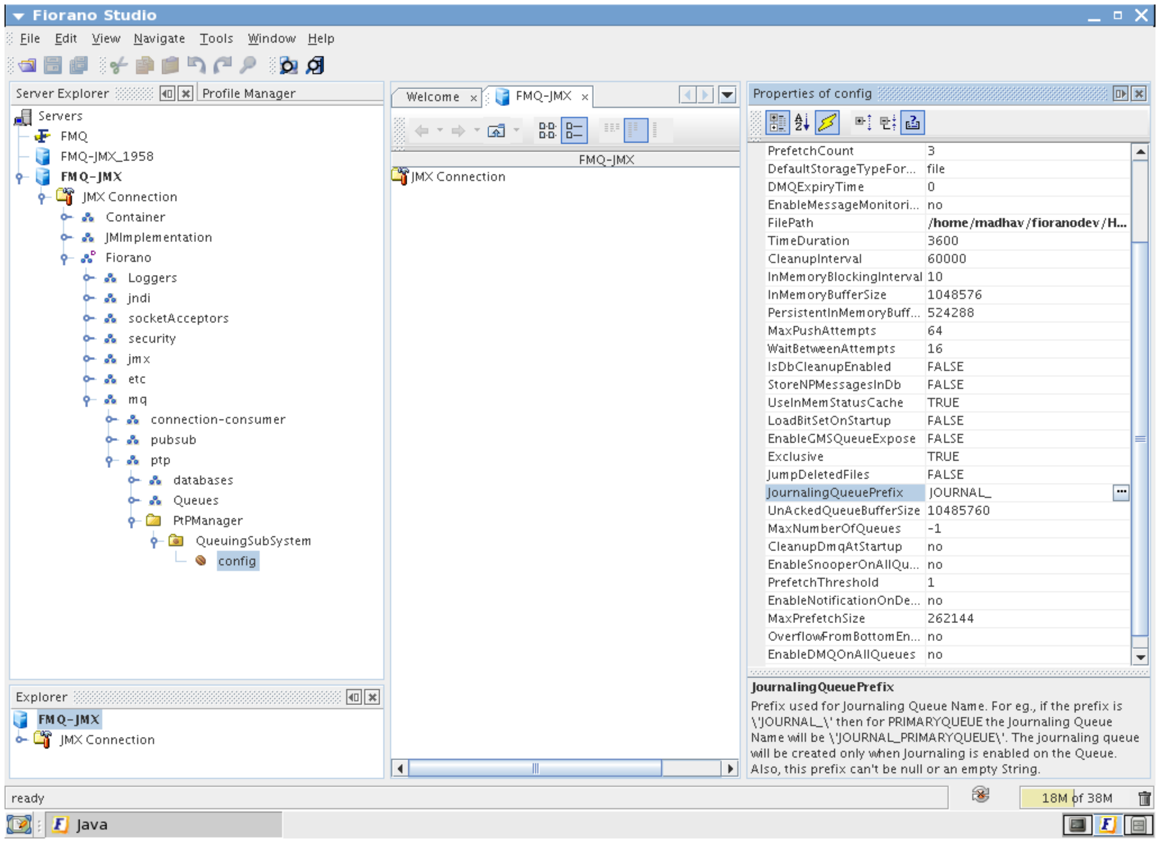
To use the **Web Management Tool**, click on **JMX > ConfigureFMQServer** and go through the path **Fiorano > mq > ptp > PtPManager > QueueingSubSystem** to click on the **Operations** tab. Call on the operation **setEnableJournalingOnQueue** and set the values for the parameters **queueName** and **enablejournaling**.

17.2.2 JournalingQueuePrefix Parameter

An additional parameter used in the **MessageJournaling**, that is, **JournalingQueuePrefix / JournalingTopicPrefix**. This is used as the prefix for a name when creating Journaling destination for a particular destination. This can be configured from Studio in both Offline and Online modes. As an example, if the prefix is JOURNAL_, and is its default value as well, then the journaling destination corresponding to the PRIMARYQUEUE will be JOURNAL_PRIMARYQUEUE.

17.2.2.1 Online mode

1. Launch **Admin Studio** and connect to a FioranoMQ Server through the **Server Explorer** pane.
2. Navigate to the **QueueingSubSystem** and select the **config** node, as shown in figure below.
3. Modify the values of the parameter **JournalingQueuePrefix**.



Properties of config

PrefetchCount	3
DefaultStorageTypeFor...	file
DMQExpiryTime	0
EnableMessageMonitori...	no
FilePath	/home/madhav/fioranodev/H...
TimeDuration	3600
CleanupInterval	60000
InMemoryBlockingInterval	10
InMemoryBufferSize	1048576
PersistentInMemoryBuff...	524288
MaxPushAttempts	64
WaitBetweenAttempts	16
IsDbCleanupEnabled	FALSE
StoreNPMessagesInDb	FALSE
UseInMemStatusCache	TRUE
LoadBitSetOnStartup	FALSE
EnableGMSQueueExpose	FALSE
Exclusive	TRUE
JumpDeletedFiles	FALSE
JournalingQueuePrefix	JOURNAL_
UnAckedQueueBufferSize	10485760
MaxNumberOfQueues	-1
CleanupDmqAtStartup	no
EnableSnooperOnAllQu...	no
PrefetchThreshold	1
EnableNotificationOnDe...	no
MaxPrefetchSize	262144
OverflowFromBottomEn...	no
EnableDMQOnAllQueues	no

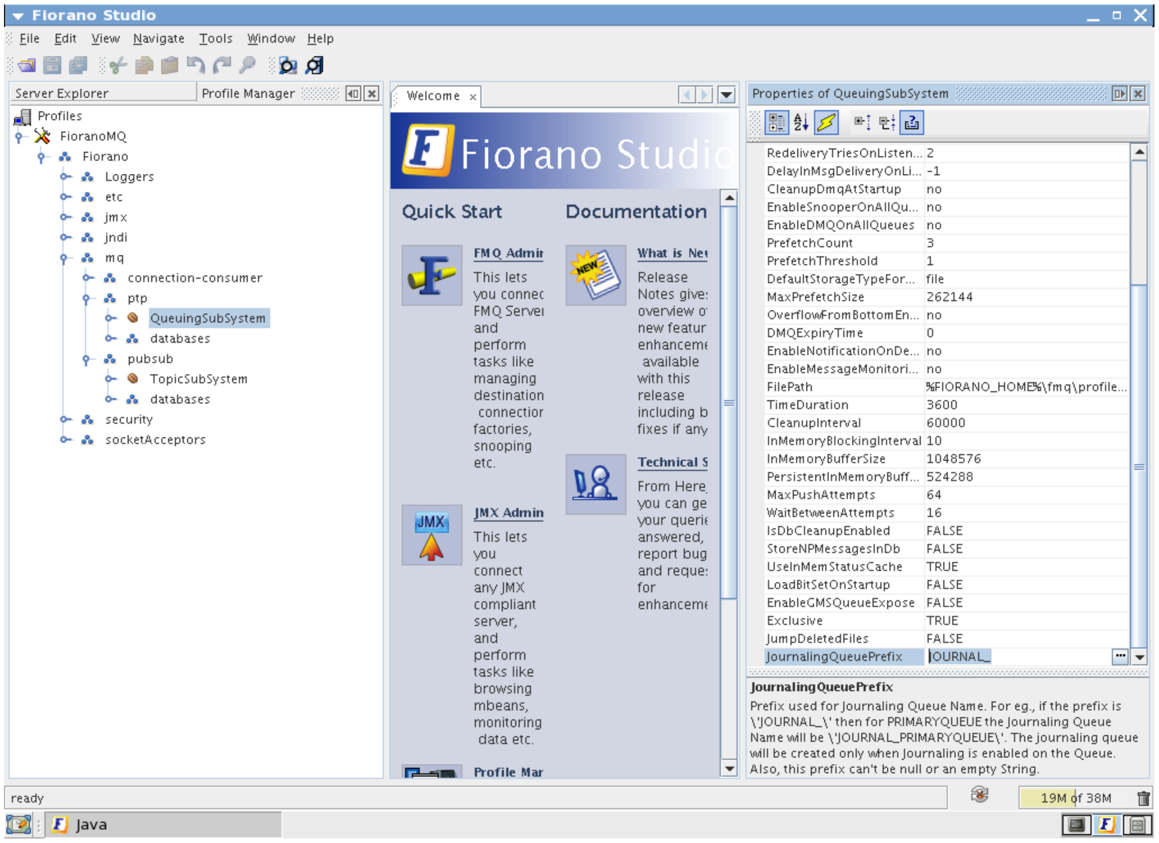
JournalingQueuePrefix

Prefix used for Journaling Queue Name. For eg., if the prefix is \'JOURNAL_\' then for PRIMARYQUEUE the Journaling Queue Name will be \'JOURNAL_PRIMARYQUEUE\'. The journaling queue will be created only when Journaling is enabled on the Queue. Also, this prefix can't be null or an empty String.

For setting the **JournalingTopicPrefix** parameter through **Studio** in the online mode login to the **FioranoMQ server** using the **FMQ-JMX** login. Go to **Fiorano > mq > pubsub > PubSubManager > TopicSubSystem > config** and set the **JournalingTopicPrefix** flag to '**JOURNAL_**'.

17.2.2.3 Offline mode

1. Launch the **Fiorano Studio** and open the **Profile Manager**. **Right-click** the **Profiles** node and select **Open Profile** from the pop-up menu. Select the required **profile** and click on the **Open** button.
2. Navigate within the component tree to select the **bean** under the **ptp > QueueingSubSystem** node.
3. In the properties panel, edit the parameter **JournalingQueuePrefix** entering the required value.
4. Navigate within the component tree to select the **bean** corresponding to **pubsub > TopicSubSystem** node.
5. In the properties panel, edit the parameter **JournalingTopicPrefix** entering the required value.
6. **Right-click** the node and select **Save** from the pop-up menu.



RedeliveryTriesOnListen...	2
DelayInMsgDeliveryOnLi...	-1
CleanupDmqAtStartup	no
EnableSnooperOnAllQu...	no
EnableDMQOnAllQueues	no
PrefetchCount	3
PrefetchThreshold	1
DefaultStorageTypeFor...	file
MaxPrefetchSize	262144
OverflowFromBottomEn...	no
DMQExpiryTime	0
EnableNotificationOnDe...	no
EnableMessageMonitori...	no
FilePath	%FIORANO_HOME%\fmq\profile...
TimeDuration	3600
CleanupInterval	60000
InMemoryBlockingInterval	10
InMemoryBufferSize	1048576
PersistentInMemoryBuff...	524288
MaxPushAttempts	64
WaitBetweenAttempts	16
IsDbCleanupEnabled	FALSE
StoreNPMessagesInDb	FALSE
UseInMemStatusCache	TRUE
LoadBitSetOnStartup	FALSE
EnableGMSQueueExpose	FALSE
Exclusive	TRUE
JumpDeletedFiles	FALSE
JournalingQueuePrefix	JOURNAL_

JournalingQueuePrefix
Prefix used for Journaling Queue Name. For eg., if the prefix is \'JOURNAL_\' then for PRIMARYQUEUE the Journaling Queue Name will be \'JOURNAL_PRIMARYQUEUE\'. The journaling queue will be created only when Journaling is enabled on the Queue. Also, this prefix can't be null or an empty String.

Note: By default, on System Topics, such as SNOOPER TOPICS (SYSTEM_MESSAGESNOOPER_QUEUE and SYSTEM_MESSAGESNOOPER_TOPIC) and Events Topic (EVENTS_TOPIC), message journaling is COMPLETELY DISABLED. The user cannot Enable/Disable the EnableJournaling flag on these topics.

Note: By default, the EnableJournaling flag is set to FALSE for each destination in the Server.

17.3 Message Journaling with HA

If using Message Journaling with HA (Replicated or Shared), the EnableJournaling flag must be enabled at both the PRIMARY as well as the SECONDARY HA servers beforehand to avoid any message loss during failovers.

This is because, when the EnableJournaling flag is enabled through online/offline mode on a queue which is in the ACTIVE server, the configurations will not be replicated to the PASSIVE server. The journaling status for the queue in the PASSIVE server will remain false. Untill Journaling is enabled for the Queue in the 'ONCE PASSIVE NOW ACTIVE' server, the messages that are sent to this queue are not 'journaled' and message losses can be observed in the Journaling Queue. Therefore, the flag must be enabled, explicitly, prior to starting the servers to avoid message losses.

This can be done by making the following changes in the Configs.cfg file of both the HA servers. This is the block defining the destination parameters for the queue 'PRIMARYQUEUE'.

ObjectName=Fiorano.mq.ptp.Queues:Name="PRIMARYQUEUE",ServiceType=Queue,type=config

ClassName=fiorano.jms.common.config.QueueConfig

EnableJournaling=true // This must be newly added/uncommented before starting servers.

ObjectName=Fiorano.mq.ptp.Queues:Name="PRIMARYQUEUE",ServiceType=Queue,type=config

17.4 Points to Remember

- When the `EnableJournaling` flag is set to true on a destination, for example SAMPLE_DEST, a new Journaling destination is created in the FioranoMQ server. All incoming messages to the SAMPLE_DEST destination are replicated to the new journaling destination.
- The file storage 'type' for this new journaling destination is dependent on the configurable parameter 'DefaultStorageTypeForQueues/DefaultStorageTypeForTopics' which is present in Queue/Topic at the Subsystem level.
    - Take the example of enabling Journaling on the queue SAMPLE_QUEUE which has the default File-based message storage. Reset the parameter from 'DefaultStorageTypeForQueues' to 'rdbms'. It may be necessary to enable the flag 'EnableRDBMS' and specify the RDBMS parameters in the profile. Please refer to Chapter 6 Configuring Message Store for more information.

      Set the flag EnableJournaling on the SAMPLE_QUEUE by following the procedure mentioned in the previous sections. This will create the Journaling Destination JOURNAL_SAMPLE_QUEUE which has the RDBMS based message store.

      Similarly, a File based storaged journaling queue can be created for a RDBMS based queue.
- By default, the EnableJournaling flag is set to 'false' for each destination in the FioranoMQ server and thus WILL NOT effect the performance of the MQ server when it is in the default mode. When the EnableJournaling flag is enabled, the performance of the MQ server is decreased significantly since this enables replicating all messages targeted on a destination to a different destination.
- Security ACLs are pre-defined for Journal Destinations based on the flag 'CreateDefaultACL' of Fiorano -> etc -> FMQConfigLoader. For more information on this parameter and on Security refer to Chapter 2 of FioranoMQ Reference Guide and Chapter 7 FioranoMQ Security. If this flag] is set to 'true', the default ACL is set is not set to 'true', no ACL is defined for the Journaling destination. They behave normally like any other newly created destination in the Server.
- Persistent as well as Non-persistent messages can be journaled to a journaling destination.
- The EnableJournaling property is basically set at the destination level in the Server. Irrespective of the 'types' of subscribers created (Durable/Non-Durable) on particular topics, the incoming messages to a Topic are journaled to a different topic, if the 'EnableJournaling' flag is set to 'true'.

- Various FioranoMQ features such as Large Message support, Context Based Routing, XA etc work as expected in case of Journaling destinations.
- When EnableJournaling is enabled on a Queue, messages will be persisted, if necessary, for the Journaling destination as well. Therefore, these messages need to be consumed immediately or the disk space will decrease at twice the normal rate compared to when EnableJournaling is not enabled.
- This feature can be used when an administrator needs to Snoop all the messages that are incoming on a destination of any ‘type’. This is different from the EnableSnooper function, which is supported on both Queues and Topics.The messages Snooped on all destinations will be replicated onto a single topic. By using this feature messages will be replicated to a destination of a similar ‘type’.

Chapter 18: Last Value Caching

18.1 Introduction

This feature is based on requirements that come from the pricing infrastructure world. To elaborate more, in the pricing world the last-value cache is based on a record with multiple fields. A pricing update message may only update at most all fields in this record, identified by keys for fields, which are analogous to updating a row in a database table. Applications registered to the trading broker receive the messages as sent, but when a new application comes along, the first thing it gets from the broker is the complete record (as cached at the time of subscription) followed by the messages updated to the record. This enables trading applications to easily build upon the latest or current trend of trading and thereon to join the stream.

In FioranoMQ context, this is analogous to - each JMS Topic can be used to store the last-value cache or a snapshot of data that can be viewed as current data and each new client subscriber application will obtain the current snapshot first and then get updates to whatever was in the snapshot on an ongoing basis. It should be noted that each update message, before being sent to the subscribers on a particular Topic on which Last-value caching is enabled is also stored in the last-value cache for that Topic.

It should be noted that the snapshot is simply a set of JMS messages, exactly as they were sent by the JMS MessageProducer.

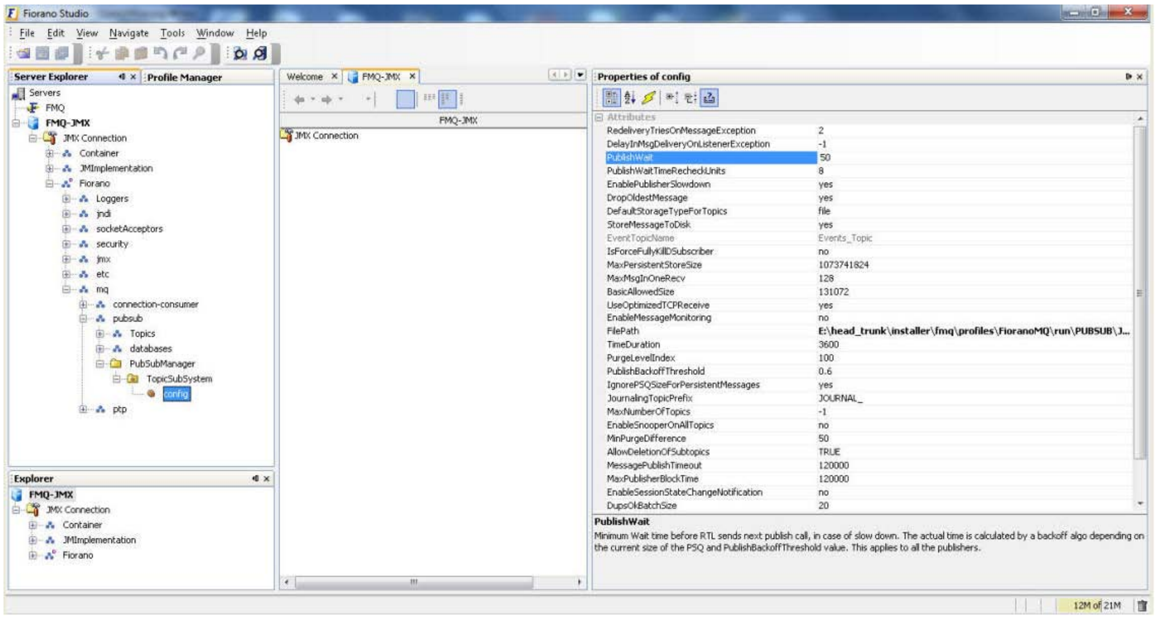
18.2 Configuring Last-value Caching

In FioranoMQ, last-value caching can be enabled on a JMS Topic. This section explains how this can be configured and thereby used along the various parameters used to enable and support last-value caching on the FioranoMQ Topics. All the related configurations for this feature are done either at the Topic Subsystem Level or at the Topic Level. Below description and corresponding figures explain how a parameter can be configured at the Topic Subsystem-Level and Topic-Level using Fiorano Studio. The same configuration can also be done from the Topics tab of the FioranoMQ Web Management Console, procedures are detailed in the corresponding chapter for Web Console.

To modify a parameter at the Topic Subsystem-Level in FioranoMQ Server Online mode, follow the steps described below:

1. Launch Fiorano Studio using the executable (bat/sh) from **%INSTALLER_HOME%/Studio/bin**.
2. Make sure that the FioranoMQ Server process is running.
3. Login to the FioranoMQ Server using **FMQ-JMX** after providing the necessary **Host Address**, **RMI Port**, **User Name** and **Password**.
4. Navigate to JMX Connection > Fiorano > mq > pubsub > PubSubManager > TopicSubSystem > **config**.

5. Select the required parameter and modify the value in the right-side parameters panel.



Attributes	
RedeliveryTriesOnMessageException	2
DelayInMsgDeliveryOnListenerException	-1
PublishWait	50
PublishWaitTimeRecheckUnits	8
EnablePublisherSlowdown	yes
DropOldestMessage	yes
DefaultStorageTypeForTopics	file
StoreMessageToDisk	yes
EventTopicName	Events_Topic
IsForceFullyKillDSubscriber	no
MaxPersistentStoreSize	1073741824
MaxMsgInOneRecv	128
BasicAllowedSize	131072
UseOptimizedTCPReceive	yes
EnableMessageMonitoring	no
FilePath	E:\head_trunk\installer\fmq\profiles\FioranoMQ\run\PUBSUB\J...
TimeDuration	3600
PurgeLevelIndex	100
PublishBackoffThreshold	0.6
IgnorePSQSizeForPersistentMessages	yes
JournalingTopicPrefix	JOURNAL_
MaxNumberOfTopics	-1
EnableSnooperOnAllTopics	no
MinPurgeDifference	50
AllowDeletionOfSubtopics	TRUE
MessagePublishTimeout	120000
MaxPublisherBlockTime	120000
EnableSessionStateChangeNotification	no
DupsOkBatchSize	20

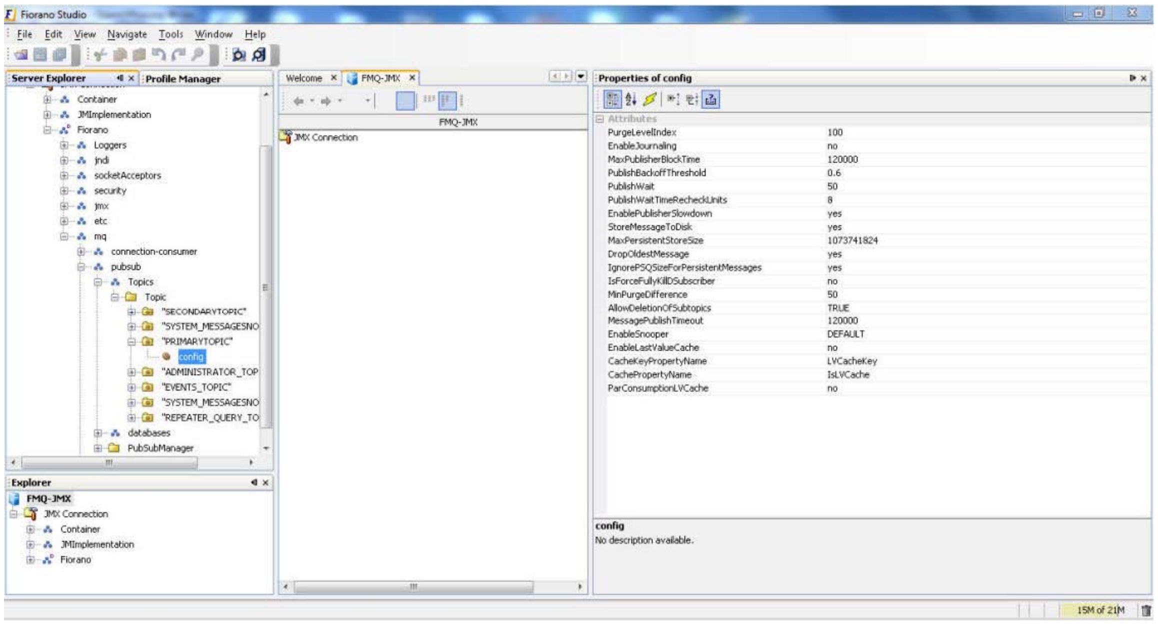
PublishWait
Minimum Wait time before RTL sends next publish call, in case of slow down. The actual time is calculated by a backoff algo depending on the current size of the PSQ and PublishBackoffThreshold value. This applies to all the publishers.

Figure: Topic Subsystem-Level Configuration

To modify a parameter at the Topic-Level in FioranoMQ Server Online mode, follow the steps described below:

1. Launch Fiorano Studio using the executable (bat/sh) from **%INSTALLER_HOME%/Studio/bin**.
2. Make sure that the FioranoMQ Server process is running.
3. Login to the FioranoMQ Server using **FMQ-JMX** after providing the necessary **Host Address**, **RMI Port**, **User Name** and **Password**.
4. Navigate to JMX Connection > Fiorano > mq > pubsub > Topics > Topic > %TOPIC_NAME% > **config**.

5. Select the required parameter and modify the value in the right-side parameters panel.



PurgeLevelIndex	100
EnableJournaling	no
MaxPublisherBlockTime	120000
PublishBackoffThreshold	0.6
PublishWait	50
PublishWaitTimeRecheckUnits	8
EnablePublisherSlowdown	yes
StoreMessageToDisk	yes
MaxPersistentStoreSize	1073741824
DropOldestMessage	yes
IgnorePSQSizeForPersistentMessages	yes
IsForceFullyKillDSubscriber	no
MinPurgeDifference	50
AllowDeletionOfSubtopics	TRUE
MessagePublishTimeout	120000
EnableSnooper	DEFAULT
EnableLastValueCache	no
CacheKeyPropertyName	LVCacheKey
CachePropertyName	IsLVCache
ParConsumptionLVCache	no

Figure – Topic-Level Configuration

18.2.1 Parameters used for Last-value Caching

18.2.1.1 EnableLastValueCache

Identifies the topic as being capable of caching a snapshot of messages which will be delivered to all new subscribers of the topic. This parameter can be configured at the Topic-Level following the way described in the earlier section of this chapter.

18.2.1.2 CacheKeyPropertyName

Message header property name that contains the Key (String) under which the message will be cached. This parameter determines the property name used to define the last-value cache Key in the JMS Message using which the message is stored in the Topic's last-value cache.

18.2.1.3 CachePropertyName

Message header property name that contains a Boolean value instructing the broker to add or remove from the cache any message under the specified Key. This parameter determines the property name used to define whether the JMS Message should be considered to be stored in the Topic's last-value cache.

18.2.1.4 ParConsumptionLVCache

This parameter determines whether to enable parallel consumption of last-value cache messages from Subscriber's copy. When the Subscriber is created, it initiates a creation of Subscriber's copy of the Topic's last-value cache. If set to false (default), the consumption of messages will start only after the Subscriber completely creates the Topic's last-value cache copy. If set to true, the time the copy is created, a parallel mechanism is enabled to consume the messages from the Subscriber's last-value cache copy.

Note: The total time taken to create the Subscriber's copy of Topic's last-value cache might be slightly higher when this parameter is set to true. This is because, in this case, the access to the Subscriber's copy is shared in mutually exclusive manner between the push and pop operations on the Queue data structure that is being used as copy.

18.2.1.5 LoadLVCIndicesAtLookup

This parameter determines the instance when the last-value cache indices are read from the file storage into the Server's in-memory buffer. These indices are stored in a Hash structure to uniquely identify the current portfolio of a particular equity. For example, if set to true, the last-value cache indices are loaded at the time of the Topic lookup and when set to false, these are loaded at the time when the MessageProducer sends the first JMS Message to the Topic.

18.2.1.6 IgnoreNullLVCKey

This parameter determines whether to ignore the message sent to a JMS Topic on which last-value cache is enabled with cache-key property not set. This key is set in the JMS Message Header as a property with name defined by CacheKeyPropertyName. If this parameter is set to true, the JMS Message will be ignored at the time of caching, but will be sent to the listening subscribers on the Topic. If this is set to false, an exception is thrown if the message has no cache-key set.

Note: This is particularly useful when sending Persistent messages on the Topic, since the Exception needs to be propagated to the JMS MessageProducer so that necessary steps can be taken accordingly.

18.2.1.7 FlushLVCDataAtStartup

This parameter determines whether to clean-up LVC data stores at the time of start-up of the FioranoMQ Server. This parameter accepts boolean values - 'true' or 'false'. Default value is 'false', which means that when the messages stored in the last-value cache are not deleted and any new Subscribers will get those messages.

Note: This flag can be used to potentially start the last-value caching afresh and to flush the data stored in the Topics' last-value cache after the Server's each run.

18.3 Using Last-value Caching

Producer Applications should be changed to use Last-value caching on a Topic. This section explains the procedure with an example. Consider a Topic named STOCK_UPDATES is configured with last-value caching enabled. The following properties are set on it:

- **EnableLastValueCache**: true
- **CacheKeyPropertyName**: (left as default LVCacheKey)
- **CachePropertyName**: (left as default IsLVCache)

The MessageProducer that currently generates STOCK_UPDATE messages is modified as below to add two new properties to each JMS Message being placed on the Topic.

1. Producer calls message.setStringProperty("LVCacheKey", "COM-1"); where "COM-1" is the identifier of the equity.
2. Producer calls message.setBooleanProperty("IsLVCache", true); using true if the holding amount is > 0, or false otherwise.

The producer can implement the rule that if holding value is 0, the equity should no longer be considered part of the portfolio. If it is > 0, then the equity is part of the portfolio. With the pseudo-code above, the last-value cache on the Topic is added to (if "COM-1" is a new holding) or updated (if a previous holding has been adjusted). If the holding was 0, the producer would have set the IsLVCache property to false, thereby causing the broker to remove COM-1 from the Topic's last-value cache.

The code block in the MessageProducer looks like this to add or adjust an entry in the Topic's last-value cache:

```
/**

...

...

 Code for creating TopicConnection, TopicSession, TopicPublisher

...

...

*/

                TextMessage textMessage = topicSession.createTextMessage();

                textMessage.setBooleanProperty("IsLVCache", true);
                textMessage.setStringProperty("LVCacheKey", "COM-A");

                textMessage.setIntProperty("CurrHoldingValue", 123);



                topicPublisher.publish(
                    textMessage,

                    javax.jms.DeliveryMode.PERSISTENT,

                    javax.jms.Message.DEFAULT_PRIORITY,

                    javax.jms.Message.DEFAULT_TIME_TO_LIVE);


...

…
```

18.4 Points to Note

- DurableSubscriber creation is disabled on a Topic on which Last-value caching is enabled. This is to ensure that when the Subscriber is started, it will only get the latest snapshot from the Topic and any incoming updates to it from that time onwards.
- Sometimes it might be necessary to compact the file storage used for the Topic's last-value cache because of its size. Generally, when the message is deleted from the FMQ database files, it is only marked as 'deleted' temporarily but it is actually removed from the hard storage when all the messages in the same file are marked as deleted. When the compaction is done, the file storage will only have the readily deliverable and will not have any messages that are already marked as deleted. For this, a JMX-based operation is provided on Topic Runtime Mbean, which can be invoked from Fiorano Studio or Web Management Console.
- No changes are required for Subscriber applications to accommodate Last-value caching changes.

Chapter 19: Message Grouping

19.1 Introduction

Let's say there is an order processing grid on 10 nodes, each of which needs to load customer account data into a cache. Each node can hold 10% of the customer data. Every node subscribes to the "ORDERS" queue at the broker. When the producer to the ORDERS queue produces a message, it sets the customer ID in the **JMSXGroupID** header. If the broker sees that this ID is unallocated, it selects a consumer from the 10 on the queue and delivers the message to it. Now ALL future messages with that ID will go to the same consumer which means it goes to the consumer with that account in its cache. This happens forever unless the consumer goes away. In that event, the broker will select a new consumer from the available ones the next time that account ID is seen in the **JMSXGroupID** header and that consumer will have to load the account data into cache.

19.2 Salient Features of Message Grouping

Messages in a message group share the same group id, i.e., they have the same group identifier property JMSXGroupID and an integer property to identify the group sequence.

Messages in a message group are always consumed by the same consumer, even if there are many consumers on a queue. They pin all messages with the same group id to the same consumer. If that consumer is closed another consumer is chosen and will receive all messages with the same group id.

A message group can be assigned to a different consumer by having the producer send a message which has "message group signal close", which is a boolean parameter, set to true. The same parameter can be used to indicate that the consumer handling this message group can be coupled with a different message group.

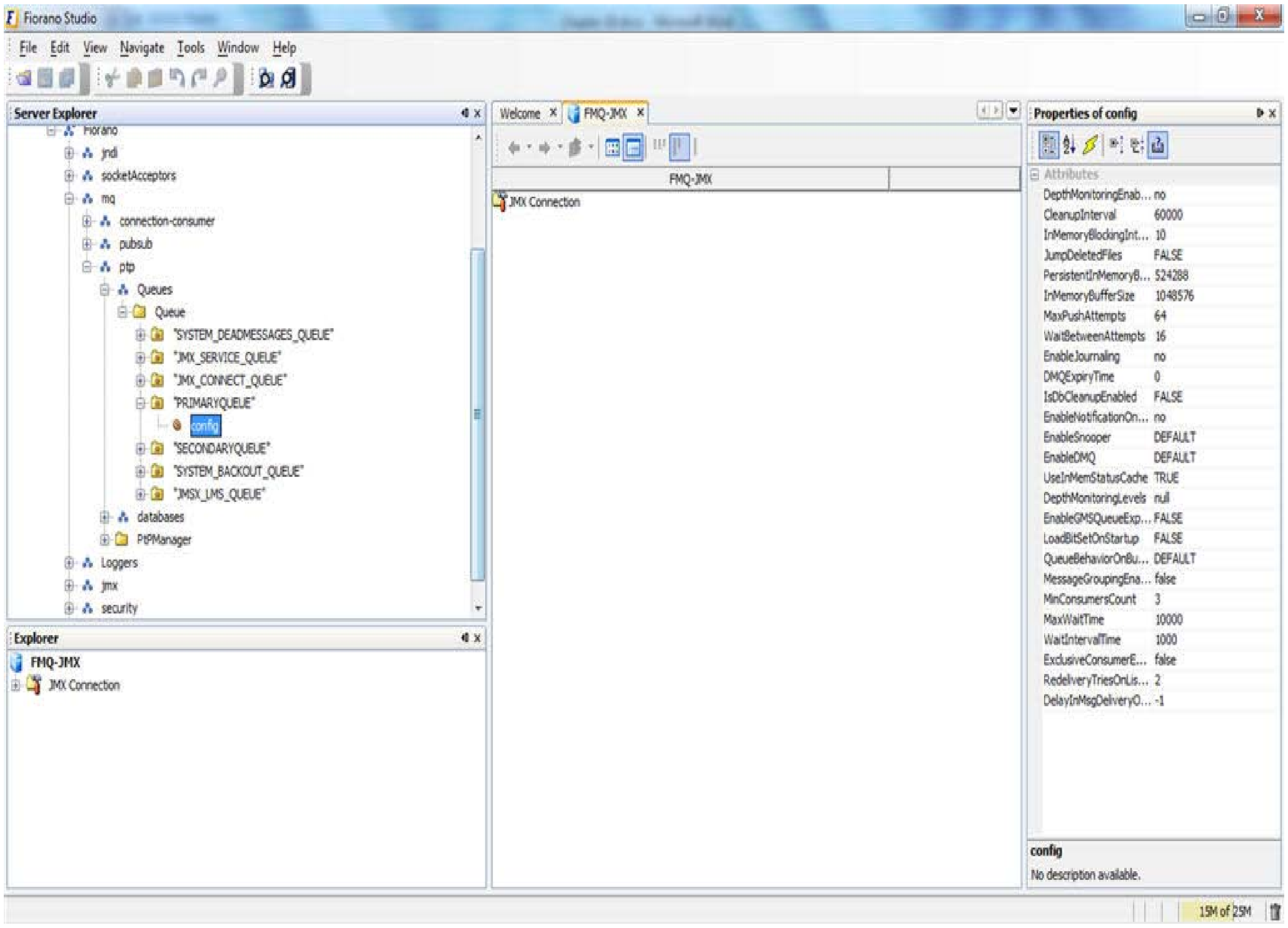
19.3 Configuring Message Grouping

In FioranoMQ, Message Grouping can be enabled on a JMS Queue. This section explains how this can be configured and thereby used along the various parameters used to enable and support Message Grouping on the FioranoMQ Queues. All the related configurations for this feature are done either at the Queue Level. Below description and corresponding figures explain how a parameter can be configured at the Queue-Level using Fiorano Studio. The same configuration can also be done from the Queues tab of the FioranoMQ Web Management Console; procedures are detailed in the corresponding chapter for Web Console.

To modify a parameter at the Queue-Level in FioranoMQ Server Online mode, follow the steps described below:

1. Launch Fiorano Studio using the executable (bat/sh) from %INSTALLER_HOME%/Studio/bin.
2. Make sure that the FioranoMQ Server process is running.

3. Login to the FioranoMQ Server using FMQ-JMX after providing the necessary Host Address, RMI Port, User Name and Password.
4. Navigate to JMX Connection > Fiorano > mq > ptp > Queues > Queue > $QUEUE_NAME > config.
5. Select the required parameter and modify the value in the right-side parameters panel.



DepthMonitoringEnab...	no
CleanupInterval	60000
InMemoryBlockingInt...	10
JumpDeletedFiles	FALSE
PersistentInMemoryB...	524288
InMemoryBufferSize	1048576
MaxPushAttempts	64
WaitBetweenAttempts	16
EnableJournaling	no
DMQExpiryTime	0
IsDbCleanupEnabled	FALSE
EnableNotificationOn...	no
EnableSnooper	DEFAULT
EnableDMQ	DEFAULT
UseInMemStatusCache	TRUE
DepthMonitoringLevels	null
EnableGMSQueueExp...	FALSE
LoadBitSetOnStartup	FALSE
QueueBehaviorOnBu...	DEFAULT
MessageGroupingEna...	false
MinConsumersCount	3
MaxWaitTime	10000
WaitIntervalTime	1000
ExclusiveConsumerE...	false
RedeliveryTriesOnLis...	2
DelayInMsgDeliveryO...	-1

19.3.1 Parameters used for Message Grouping

19.3.1.1 MessageGroupingEnabled

This parameter determines the Queue behavior for MessageGrouping. If enabled, messages belonging to the same group identified by the property JMSXGroupID will be sent to the same consumer.

19.3.1.2 MinConsumersCount

Determines the minimum number of consumers on the queue before the message groups are distributed between them.

19.3.1.3 MaxWaitTime

Determines the maximum wait time before the message groups are distributed among the existing consumers on a queue.

19.3.1.4 WaitIntervalTime

Determines the wait interval time in case of Message Grouping, after which the minimum number of consumers or maximum wait time is checked until one of the conditions is satisfied.

19.4 Using Message Grouping

Fiorano Queues should be configured with the parameter, MessageGroupingEnabled to denote that the messages that arrive onto this destination should be grouped and sent to the consuming applications. And from the client application point of view, JMS producers should be changed to fill in the JMSXGroupID message header with some String value of any choice and a message sequence property JMSXGroupSeq with positive integer values e.g.

```
MessageProducer producer = session.createProducer(“sample_destination”);

Message message = session.createTextMessage("sample_text");

message.setStringProperty("JMSXGroupID", "sample_group_index");

message.setIntProperty("JMSXGroupSeq", 0); //1, 2, 3...

producer.send(message);
```

In order to close a message group, you can add a negative sequence number in the JMS producer. A sequence close signal should be on the LAST message of a group and the message is sent to the original owner

```
MessageProducer producer = session.createProducer(“sample_destination”);

Message message = session.createTextMessage("sample_text");

message.setStringProperty("JMSXGroupID", "sample_group_index");

message.setIntProperty("JMSXGroupSeq", -1); // used as int property here

producer.send(message);
```

This will close the message group so if another message is sent in the future with the same message group ID it will automatically be re-assigned to a new consumer.

19.4.1 Preferred Groups

In addition to the above mentioned Message grouping, the consuming applications can optionally indicate the list of message groups it might be interested in, so that the server will assign those groups to it, if they weren't assigned already to any other consumer. This should be included in String format while creating a message consumer (in place of MessageSelector expression) and understandable by the FioranoMQ Server such that it can parse the expression and note the MessageGroups that the MessageConsumer is interested in. The preferred groups should be mentioned in the following way:

```
// Consumer 1 is interested in Odd-numbered groups

String preferredGroupsExpr1 =

“jms_fiorano_preferred_message_groups=

MGROUP_1; MGROUP_3; MGROUP_5”;



// Consumer 2 is interested in Even-numbered groups

String preferredGroupsExpr2 =

“jms_fiorano_preferred_message_groups=

MGROUP_2; MGROUP_4; MGROUP_6”;



// Create Consumer #1

javax.jms.QueueSession queueSession1 =

queueConnection.createQueueSession(false, javax.jms.Session.

AUTO_ACKNOWLEDGE);

javax.jms.QueueReceiver queueReceiver1 =

queueSession1.createReceiver(queue, preferredGroupsExpr1);



// Create Consumer #2

javax.jms.QueueSession queueSession2 =

                    queueConnection.createQueueSession(false, javax.jms.Session.

                                                           AUTO_ACKNOWLEDGE);

javax.jms.QueueReceiver queueReceiver2 =
queueSession2.createReceiver(queue, preferredGroupsExpr2);
```

As mentioned above, each “Preferred Group” is separated by the semi-colon (;) character.

This mechanism is particularly helpful when working with High Availability when the Consuming applications fail-over to the Backup HA Server. In the event of Primary Server failure, the Backup server takes over and triggers the consuming applications to failover their corresponding JMS Connections to the Backup server. In case Preferred Groups are set for a QueueReceiver, the MessageGroups which were handled by a consumer before the PRIMARY SERVER FAILURE event were re-assigned to the same consumer even after the Backup Server takes over.

Chapter 20: Message Encryption

20.1 Key Generation

To generate a key, FioranoMQ comes bundled with a utility class called EncryptorImpl. This utility class allows applications to create keys for specified algorithms.

The methods within this utility class that support this function are given below:

20.1.1 public String generateKey (String algoName)

This API can be used to generate a key as shown below:

```
EncryptorImpl em = new EncryptorImpl ();

String key = em.generateKey (“DES”);
```

Using this class requires adding the statement below to the application:

```
import fiorano.jms.services.msg.Encryption;
```

20.2 Per Message Encryption

FioranoMQ provides ‘per message’ encryption that allows JMS applications to selectively encrypt messages before distributing them over the network.

Support for encryption has been added to the FioranoMessage class. APIs added to the FioranoMessage class are:

20.2.1 public void enableEncryption ()

Throws a FioranoException.

This is used to encrypt a message prior to distributing it over the network. DES is the default algorithm used.

20.2.2 public void enableEncryption (String algo, String key)

Throws a FioranoException.

This API views the algorithm and the key as string parameters to be used to encrypt messages to be distributed over the network.

20.2.3 public void decrypt ()

Throws a FioranoException.

This API is used to decrypt received messages using the default algorithm.

20.2.4 public void decrypt (String algo, String key)

Throws a FioranoException.

This API is used to decrypt received messages using the same algorithm and key that was provided during the encryption of the message.

20.3 Per Destination Encryption

A destination (topic or queue) can be marked as encrypted at the time of their creation. All the messages that are sent to this destination will be encrypted. The encryption information is maintained in the metadata associated with the destination. For this purpose, the following APIs have been added to the TopicMetaData and QueueMetaData classes:

20.3.1 public void setEncryption ()

Throws a FioranoEncryption.

This API encrypts messages intended for the destination using the default encryption algorithm.

20.3.2 public void setEncryption (String algo, String key)

Throws a FioranoEncryption.

This API encrypts messages intended for the destination by using the input parameters of the encryption algorithm and the encryption key.

An application program does not need to explicitly decrypt a message associated with a destination. Messages are delivered in decrypted form to all subscribing applications.

20.4 Note on Installation and Samples

The FioranoMQ installation comes bundled with samples that illustrate the use of encryption and decryption.

These samples are located in the fmq\samples\PubSub\MessageEncryption and fmq\samples\PTP\MessageEncryption directories within the FioranoMQ installation.

It is recommended that these samples be run to understand the encryption and the decryption support provided by FioranoMQ.

For message encryption support the file cryptix.jar must be present in the classpath while using encryption in JMS client applications. This file comes bundled within FioranoMQ 7.1 and later versions.

The file java.security file located in \j2sdk1.4.0\jre\lib\security must be modified and the following should be appended to the file:

Security.provider.n=cryptix.provider.Cryptix

Chapter 21: Message Compression

21.1 Per Message Compression

In per message Compression, clients can enable or disable compression for each message. This function has been added to the FioranoMessage class.

Relevant public methods are listed below:

21.1.1 public void enableCompression()

Message compression is enabled for a message using the default compression level and the default compression strategy.

21.1.2 public void enableCompression (int compressionLevel, int compressionStrategy)

Message compression is enabled for a message with the required compression level and strategy provided as input parameters.

Note - Please refer to Message Compression Specifications for an explanation of the possible values for compression levels and compression strategies. Constants are available in the Fiorano.jms.services.IFioranoConstants class and have to be imported into the client application to be used.

21.1.3 public int getCompressionRatio ()

This is used to retrieve the compression ratio (expressed as a percentage) achieved after the message has been sent. The computation for the ratio is: :

Compression Ratio = ((uncomp – comp)/comp) * 100

'Uncomp' represents the uncompressed message body size and 'comp' represents the compressed message body size.

Depending on this ratio, clients have the option of deciding the optimum compression level and strategy for particular scenarios. This value is available after the send/publish call returns and also after a message is received.

21.1.4 public void setCompressionLevel (int compressionLevel)

This API is used to set the compression level for a message.

21.1.5 public void setCompressionStrategy (int compressionStrategy)

This API is used to set the compression strategy for a message.

21.1.6 public int getCompressionLevel ()

This API retrieves the compression level as an integer value.

21.1.7 public int getCompressionStrategy ()

This API retrieves the compression strategy as an integer value.

Samples illustrating message compression are bundled within FioranoMQ. Please refer to the /MessageCompression/PerMessage directory of the /fmq/samples/PubSub and /fmq/samples/PTP directory of the FioranoMQ installation directory/package. It is recommended that these samples be run to understand the usage of the APIs better.

21.2 Per Destination Compression

Compression on a per destination basis requires that all messages sent on a particular destination be compressed using the compression level and strategy specified by the client.

Compression support on a per destination basis is provided through enabling compression in the corresponding metadata used to create a destination on the Admin Connection such as in the TopicMetadata and the QueueMetadata classes.

The methods listed below exist in these classes to support compression:

21.2.1 public void enableCompression ()

This API enables message compression on a topic/queue using the default compression level and strategy.

21.2.2 public void enableCompression (int level, int strategy)

This API enables message compression on a topic/queue using the input parameters of the compression level and strategy.

To change the compression level or strategy for messages on the destination on which compression has been enabled, a client application can call upon the setCompressionLevel () and setCompressionStrategy () APIs of the FioranoMessage class, to perform this function.

Samples illustrating message compression on a per destination basis are bundled within FioranoMQ. Please refer to the /MessageCompression/PerDestination directory of the /fmq/samples/PubSub and /fmq/samples/PTP directory of the FioranoMQ installation directory/package. It is recommended that these samples be run to understand the usage of the APIs better.

21.3 Message Decompression

FioranoMQ handles decompression of messages internally. Client applications do not have to explicitly use any APIs to decompress a compressed message. At runtime, decompression is automatically performed if the message is compressed and then delivered to the message consumers.

21.4 Proprietary Compression Implementation Plug-in Support

For providing plug-in support for proprietary compression implementations, APIs within the public interface Fiorano.jms.services.msg.compression.ICompressionManager have to be implemented. The default CompressionManager interface used by FioranoMQ implements the APIs within this interface. The APIs within the CompressionManager interface are listed below.

21.4.1 public byte[] compress(byte[] input)

This method returns the compressed byte array for the input byte array.

21.4.2 public byte[] decompress(byte[] input)

This method returns the decompressed byte array from the input compressed byte array.

21.4.3 public void setCompressionLevel (int level)

This method sets the compression level that is to be used by the CompressionManager instance.

21.4.4 public void getCompressionLevel ()

This method retrieves the compression level that is being used by the CompressionManager instance.

21.4.5 public void setCompressionStrategy (int strategy)

This method sets the compression strategy, provided as an input parameter, that is to be used by the CompressionManager instance.

21.4.6 public void getCompressionStrategy ()

This method retrieves the compression strategy being used by the CompressionManager instance.

21.4.7 public float getCompressionRatio ()

This method retrieves the ratio of the compression achieved, which isexpressed as a percentage.

The proprietary CompressionManager interface is called upon by setting its name in the ConnectionFactoryMetadata class. The ConnectionFactoryMetadata class is used to create new ConnectionFactory objects using the Admin Connection. The proprietary CompressionManager implementation is applied to messages sent on connections that have been created using the ConnectionFactory object for which proprietary ConnectionManager is set.

For more information about the methods present in ConnectionFactoryMetadata, refer to the ConnectionFactoryMetadata class present in the Java docs.

21.4.8 public void setCompressionManager (string Manager)

This API sets the CompressionManager class name that is to be used for message compression . The default value of the CompressionManager is Fiorano.jms.services.msg.compressionManagerImpl.

For example, if the new implementation is named Fiorano.jms.services.compression.mycomp.MyCompressionManager, the above API for a connection factory metadata object cfMD will be called upon using:

cfMD.setCompressionManager
(“Fiorano.jms.services.compression.mycomp.MyCompressionManager”).

21.4.9 public string getCompressionManager ()

This API is used to retrieve the CompressionManager instance that is being used by a Connection Factory object.

Chapter 22: Logger

FioranoMQ incorporates tracing and logging facilities for easy detection of errors in the messaging system. The FioranoMQ Administrator can dynamically set different tracing levels for each individual FioranoMQ components.

22.1 Offline Configuration

The path of log files and the log levels of the server instance have to be set in the current profile of the server. These properties can be located in the file Configs.xml located under the 'conf' directory of each profile, i.e., ***$Fiorano_Home/fmq/profiles/<profileName>/conf***

Examples:

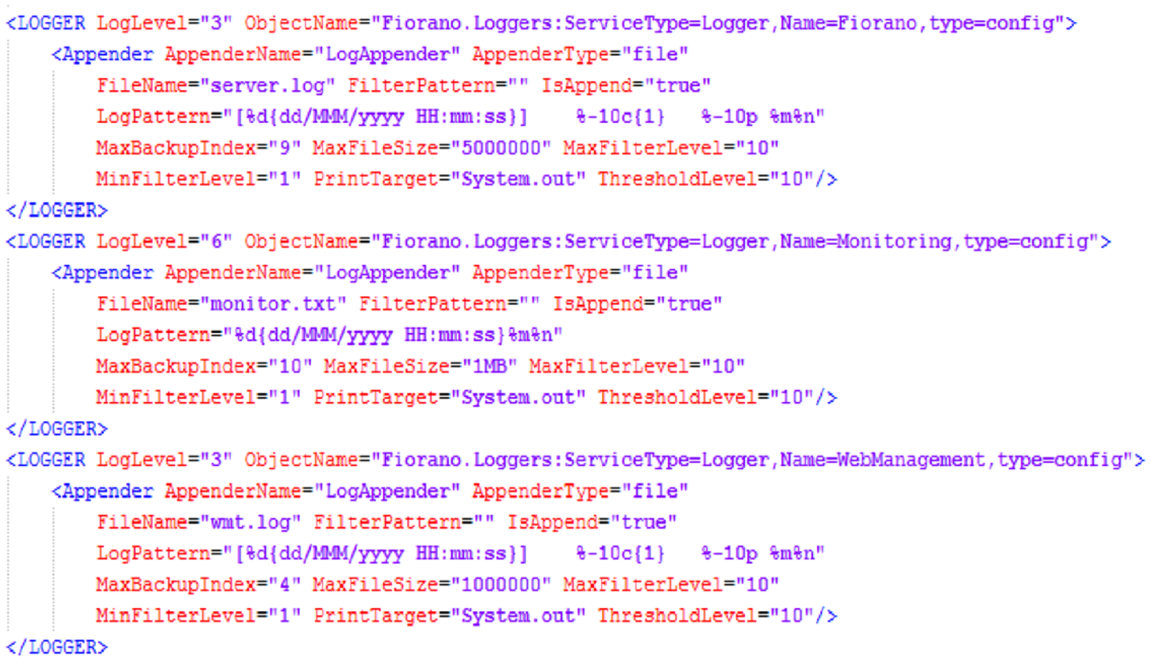
• $Fiorano_Home/fmq/ profiles/FioranoMQ/ conf/Configs.xml

• $Fiorano_Home/ fmq/profiles/FioranoMQ_HA_rpl/HAPrimary/conf/Configs.xml

The various log levels of the Server are:

- Fatal - 1
- Error - 2
- Warning - 3
- Info - 4
- Debug - 5
- Trace - 6

The following figure illustrates a section in Configs.xml which contains the Logger node.



```
<LOGGER LogLevel="3" ObjectName="Fiorano.Loggers:ServiceType=Logger,Name=Fiorano,type=config">
    <Appender AppenderName="LogAppender" AppenderType="file"
        FileName="server.log" FilterPattern="" IsAppend="true"
        LogPattern="[%d{dd/MMM/yyyy HH:mm:ss}]    %-10c{1}   %-10p %m%n"
        MaxBackupIndex="9" MaxFileSize="5000000" MaxFilterLevel="10"
        MinFilterLevel="1" PrintTarget="System.out" ThresholdLevel="10"/>
</LOGGER>
<LOGGER LogLevel="6" ObjectName="Fiorano.Loggers:ServiceType=Logger,Name=Monitoring,type=config">
    <Appender AppenderName="LogAppender" AppenderType="file"
        FileName="monitor.txt" FilterPattern="" IsAppend="true"
        LogPattern="%d{dd/MMM/yyyy HH:mm:ss}%m%n"
        MaxBackupIndex="10" MaxFileSize="1MB" MaxFilterLevel="10"
        MinFilterLevel="1" PrintTarget="System.out" ThresholdLevel="10"/>
</LOGGER>
<LOGGER LogLevel="3" ObjectName="Fiorano.Loggers:ServiceType=Logger,Name=WebManagement,type=config">
    <Appender AppenderName="LogAppender" AppenderType="file"
        FileName="wmt.log" FilterPattern="" IsAppend="true"
        LogPattern="[%d{dd/MMM/yyyy HH:mm:ss}]    %-10c{1}   %-10p %m%n"
        MaxBackupIndex="4" MaxFileSize="1000000" MaxFilterLevel="10"
        MinFilterLevel="1" PrintTarget="System.out" ThresholdLevel="10"/>
</LOGGER>
```

There are two Logger Nodes present in the Configs.xml. Each Logger has one appender called 'LogAppender'. These appenders are file based Appenders. The log level can be changed by changing the LogLevel attribute. The default log level of the logger has the objectName - Fiorano is 3. The default log level of the logger has the objectName - Monitoring is 6. The default names of the log files onto which the appenders log data, are server.log, and monitor.txt.

The default location of these files is $Fiorano_Home/fmq/profiles/<profile_name>/run/logs where profile_name is the profile on which the server is run.

Note: If the Server is running in a shared HA profile, the logs directory gets created in the path of the directory specified as the -dbPath parameter.

Example:

```
fmq.bat/sh –profile FioranoMQ_HA_shared/HAPrimary  –dbPath /home/Fiorano/db

fmq.bat/sh –profile FioranoMQ_HA_shared/HASecondary  –dbPath /home/Fiorano/db
```

Here the logs directory gets created in the location /home/Fiorano/db. Please note that in the instance of Shared HA that uses the above commands, both the Primary and the Secondary servers log their data to the same location.

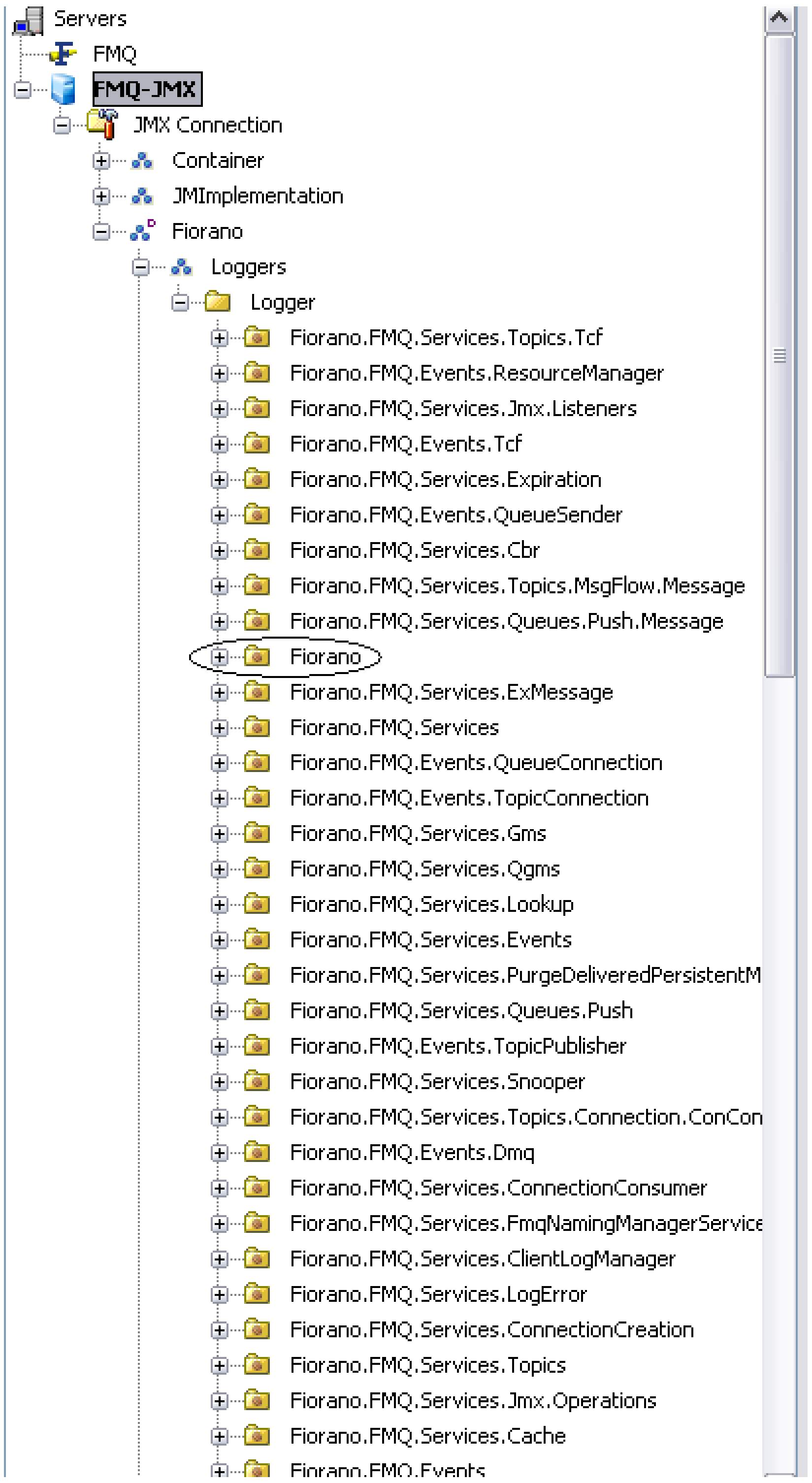
If the absolute path is not specified the log files are created in the location mentioned above. You can provide an absolute path to the FileName attribute like /home/Fiorano/server.log, in which case the log file is created in the path specified.

Note: The changes, above, have to be made offline.

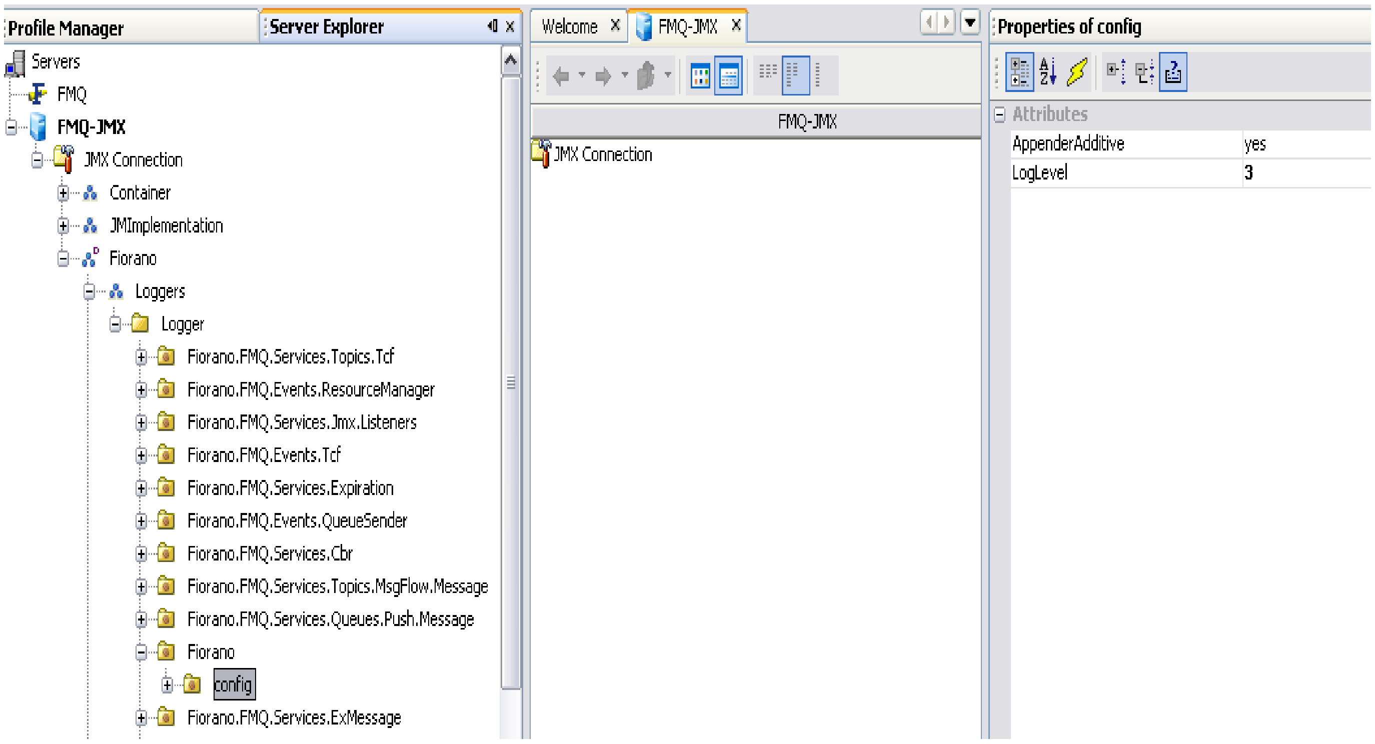
22.2 Online Configuration

Modifying log levels for each logger under 'Fiorano' can also be done while the server is running. This doesn't require a server restart to get them affected and one can do these operations by logging into FMQ Server through JMX. JMX login can be done in two ways: through Fiorano Studio which provides the login in the server explorer tab and through logging in through the web console (WMT).

The logger structure, below, appears in Fiorano Studio once a user logs in through FMQ-JMX:


Servers
FMQ
FMQ-JMX
JMX Connection
Container
JMImplementation
Fiorano
Loggers
Logger
Fiorano.FMQ.Services.Topics.Tcf
Fiorano.FMQ.Events.ResourceManager
Fiorano.FMQ.Services.Jmx.Listeners
Fiorano.FMQ.Events.Tcf
Fiorano.FMQ.Services.Expiration
Fiorano.FMQ.Events.QueueSender
Fiorano.FMQ.Services.Cbr
Fiorano.FMQ.Services.Topics.MsgFlow.Message
Fiorano.FMQ.Services.Queues.Push.Message
Fiorano
Fiorano.FMQ.Services.ExMessage
Fiorano.FMQ.Services
Fiorano.FMQ.Events.QueueConnection
Fiorano.FMQ.Events.TopicConnection
Fiorano.FMQ.Services.Gms
Fiorano.FMQ.Services.Qgms
Fiorano.FMQ.Services.Lookup
Fiorano.FMQ.Services.Events
Fiorano.FMQ.Services.PurgeDeliveredPersistentM
Fiorano.FMQ.Services.Queues.Push
Fiorano.FMQ.Events.TopicPublisher
Fiorano.FMQ.Services.Snooper
Fiorano.FMQ.Services.Topics.Connection.ConCon
Fiorano.FMQ.Events.Dmq
Fiorano.FMQ.Services.ConnectionConsumer
Fiorano.FMQ.Services.FmqNamingManagerService
Fiorano.FMQ.Services.ClientLogManager
Fiorano.FMQ.Services.LogError
Fiorano.FMQ.Services.ConnectionCreation
Fiorano.FMQ.Services.Topics
Fiorano.FMQ.Services.Jmx.Operations
Fiorano.FMQ.Services.Cache
Fiorano.FMQ.Events

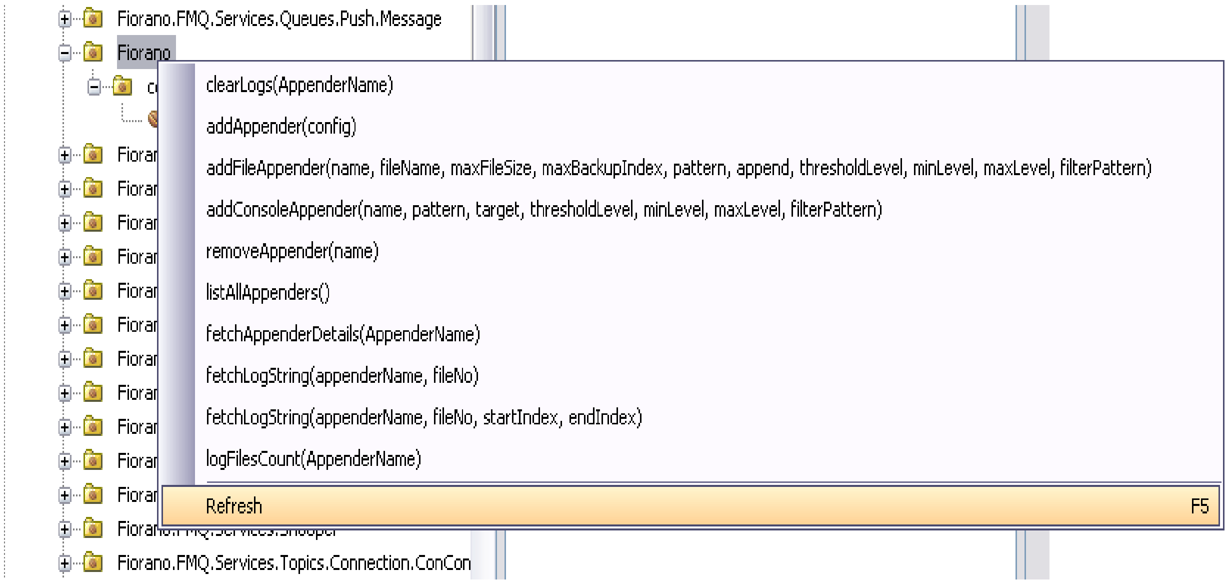
Select the 'Fiorano' logger (circled above) and expand it by clicking on the config node. In the properties window change the log level of the logger, as shown in the figure below.


Profile Manager
Server Explorer
Servers
FMQ
FMQ-JMX
JMX Connection
Container
JMImplementation
Fiorano
Loggers
Logger
Fiorano.FMQ.Services.Topics.Tcf
Fiorano.FMQ.Events.ResourceManager
Fiorano.FMQ.Services.Jmx.Listeners
Fiorano.FMQ.Events.Tcf
Fiorano.FMQ.Services.Expiration
Fiorano.FMQ.Events.QueueSender
Fiorano.FMQ.Services.Cbr
Fiorano.FMQ.Services.Topics.MsgFlow.Message
Fiorano.FMQ.Services.Queues.Push.Message
Fiorano
config
Fiorano.FMQ.Services.ExMessage
Welcome
FMQ-JMX
FMQ-JMX
JMX Connection
Properties of config
Attributes

AppenderAdditive	yes
LogLevel	3

After modifying the log level, save the configurations by right clicking on ‘FMQ-JMX’ node.

Several other operations can be done each logger node, for instance like adding a file appenders, console appenders, list all appenders, removing appenders etc. For instance right click on the ‘Fiorano’ logger node and you choose from the list of operations as shown below:


Fiorano.FMQ.Services.Queues.Push.Message
Fiorano
clearLogs(AppenderName)
addAppender(config)
addFileAppender(name, fileName, maxFileSize, maxBackupIndex, pattern, append, thresholdLevel, minLevel, maxLevel, filterPattern)
addConsoleAppender(name, pattern, target, thresholdLevel, minLevel, maxLevel, filterPattern)
removeAppender(name)
listAllAppenders()
fetchAppenderDetails(AppenderName)
fetchLogString(appenderName, fileNo)
fetchLogString(appenderName, fileNo, startIndex, endIndex)
logFilesCount(AppenderName)
Refresh
F5
Fiorano.FMQ.Services.Topics.Connection.ConCon

The detailed description of these operations and executing them from Web console (WMT) can be found in **Chapter 8 Logger Configuration** of *FioranoMQ Reference Guide*.

22.3 Fiorano Client Logger

FioranoMQ Client Logger uses SLF4J which serves as a simple facade or abstraction for various logging frameworks, such as java.util.logging, logback and log4j. So any logging implementation can be plugged in as per customer's preference. For this to work, slf4j-api-<version>.jar along with the jars used by the chosen underlying logging framework (log4j, Logback, JUL, etc) needs to be placed in the application's classpath. By default, slf4j-api-1.7.5.jar is shipped and its classpath is mentioned in run-client.conf under %FIORANO_HOME%\fmq\bin. To switch logging frameworks, just replace slf4j bindings on your class path. For example, to switch from java.util.logging to log4j, just replace slf4j-jdk14-1.7.5.jar with slf4j-log4j12-1.7.5.jar.

In order to use log4j framework, slf4j-api-<version>.jar along with slf4j-log4j12-<version>.jar and log4j.jar need to be placed in the java classpath. Loggers can be configured using log4j.properties. By default, the log4j.properties file can be located under %FIORANO_HOME%\fmq\bin.However, the log4j.properties can be placed anywhere and the path should be given under <java.classpath>. If the log4j.properties is in D:\Logger_properties\, then D:\Logger_properties\ should be mentioned in the classpath. <java.classpath> can be mentioned using " -cp " command in javac/java (or) can be given in run-client.conf present under %FIORANO_HOME%\fmq\bin.

Examples:

1. javac -cp D:\Logger_properties\ app_name.java
2. java -cp D:\Logger_properties\ app_name

Parent-child relationship in FioranoMQ Client Logger:

FioranoClientLogger follows a naming hierarchy. The logger named "log4j.logger.Fiorano.FMQ.Services.ClientRootLoggerServices.FMQClientLoggerServices" is the parent of the logger named log4j.logger.Fiorano.FMQ.Services.ClientRootLoggerServices.FMQClientLoggerServices.Admin". The inherited level for a given logger say logger_name, is equal to the first non-null level in the logger hierarchy, starting at logger_name and proceeding upwards in the hierarchy towards the root logger. If logger named "log4j.logger.Fiorano.FMQ.Services.ClientRootLoggerServices.FMQClientLoggerServices.Admin" is not assigned any log level, then it will inherit its level from "log4j.logger.Fiorano.FMQ.Services.ClientRootLoggerServices.FMQClientLoggerServices" (provided this logger has been assigned a level. Otherwise it will inherit from its parent logger and so on).

A logging request is said to be enabled if its level is higher than or equal to the level of its logger. Otherwise, the request is said to be disabled. The levels are as follows: ALL < DEBUG < INFO < WARN < ERROR < FATAL < OFF.

Note: For more information on Appender, MaxFileSize, MaxBackupIndex, Append, Levels and layout, please refer to **Chapter 8 Logger Configuration** in *FioranoMQ Reference Guide*.

Chapter 23: XA

Note:

- Add the corresponding JAR files for the database configured for the FioranoMQ server in the Configuration file of the server: *fmq.conf* present under fmq_installation_dir\fmq\bin.
- Start the FioranoMQ server with one of the database, you have to run *create-database.bat/sh* after adding the required JAR in the classpath of *create-database.conf.*

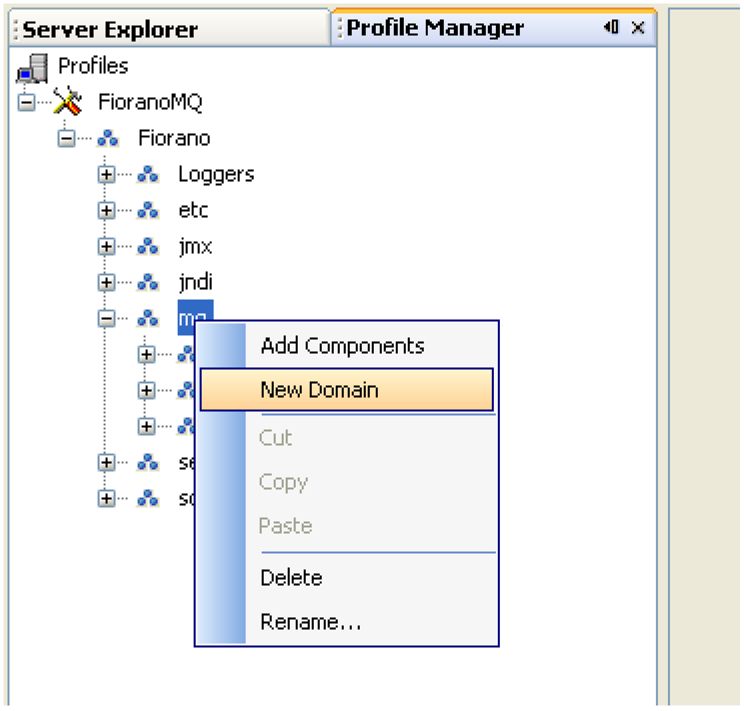
23.1 How to Enable XA

Note: XA c can be configured in offline mode only.

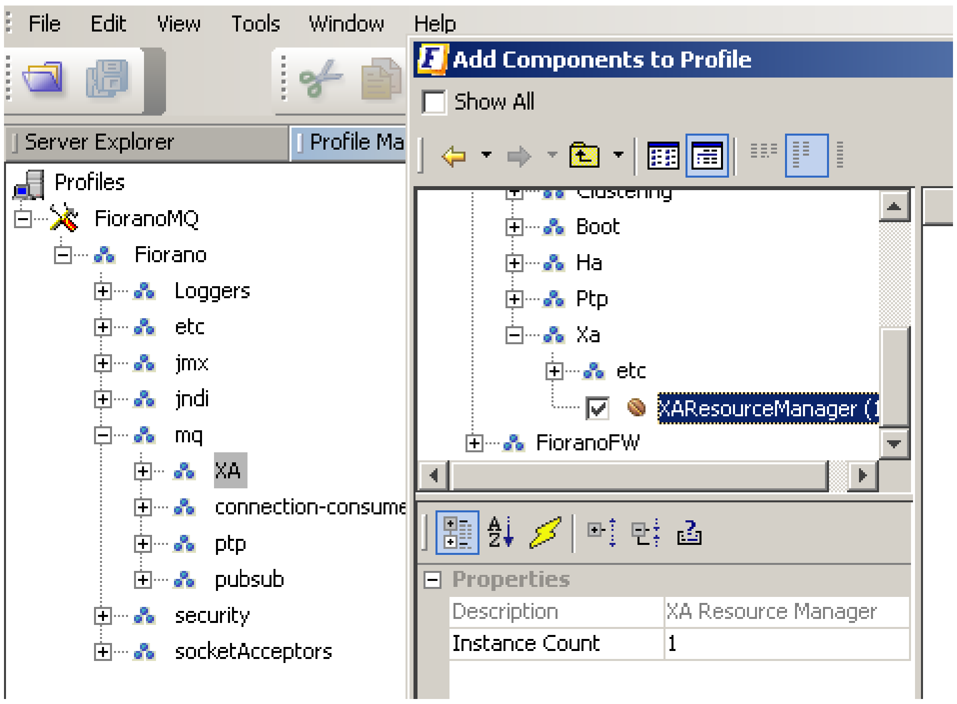
For information about configuring profiles through a text based file, see FioranoMQ Getting Started.

FioranoMQ comes with a preconfigured XA enabled FioranoMQ profile named FioranoMQ_XA. XA can be enabled in the default profile by following the steps given below:

1. Launch Fiorano Studio and open **FioranoMQ** (default profile) using the **ProfileManager**
2. Navigate **to FioranoMQ>Fiorano>mq**. Right-click and select **New Domain** from the pop-up menu.

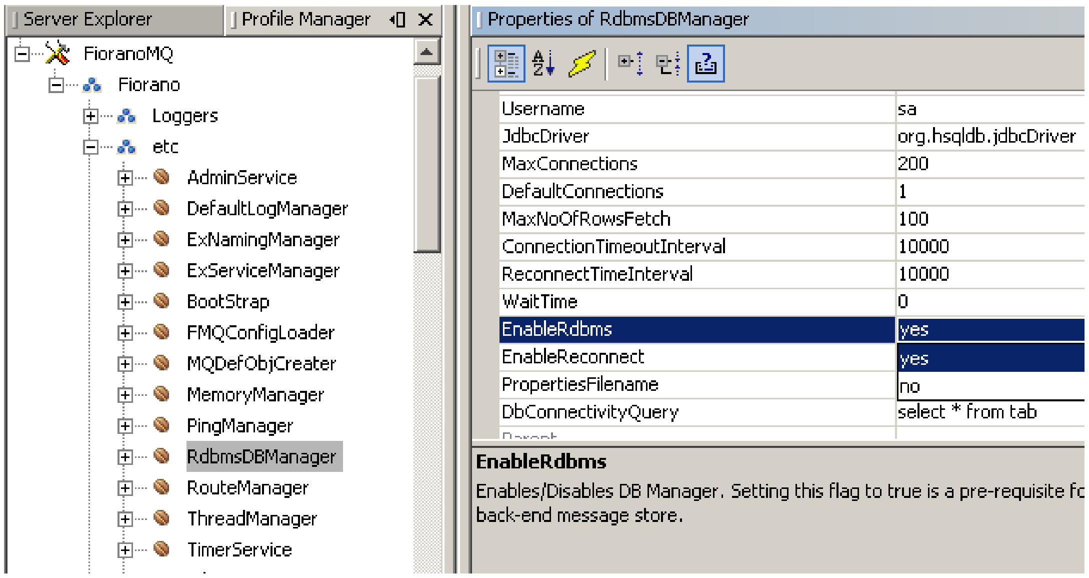


3. Enter the domain name in the **Input** pane displayed and click on the **OK** button. This domain is named XA in the example **.**
4. Right-click the new domain and select **Add Components** from the pop-up menu. An **Add Components to the Profile** window is displayed.



Description	XA Resource Manager
Instance Count	1

5. Navigate to **Fiorano>Jms>XA>etc** and check the **XAResourceManager** component and click on the **OK** button.
6. Resolve all the unresolved dependencies (which are marked with a RED icon). Select the unresolved dependencies. In the property pane choose the right value for this service instance.
7. Navigate to **Fiorano>Fiorano>mq>XA> FileDBManager> FileDBManager8** and in the **Properties of FileDBManager8** enter the path name **XA**. XA requires RDBMS based storage. Enable this as shown below:



Username	sa
JdbcDriver	org.hsqldb.jdbcDriver
MaxConnections	200
DefaultConnections	1
MaxNoOfRowsFetch	100
ConnectionTimeoutInterval	10000
ReconnectTimeInterval	10000
WaitTime	0
EnableRdbms	yes
EnableReconnect	yes
PropertiesFilename	no
DbConnectivityQuery	select * from tab

EnableRdbms
Enables/Disables DB Manager. Setting this flag to true is a pre-requisite fc
back-end message store.

8. Add the XAResourceManager component to the dependency list of **Fiorano -> etc -> ExServiceManager**. This can be done by right-clicking the **Dependson** node under the **ExServiceManager.** An **Add Component as Dependency** dialog box is displayed**.** Check **XAResourceManager** and click on the **OK** button**.**
9. Right-click the profile and select the **Save** option from the pop-up menu.

23.2 XA Prerequisites

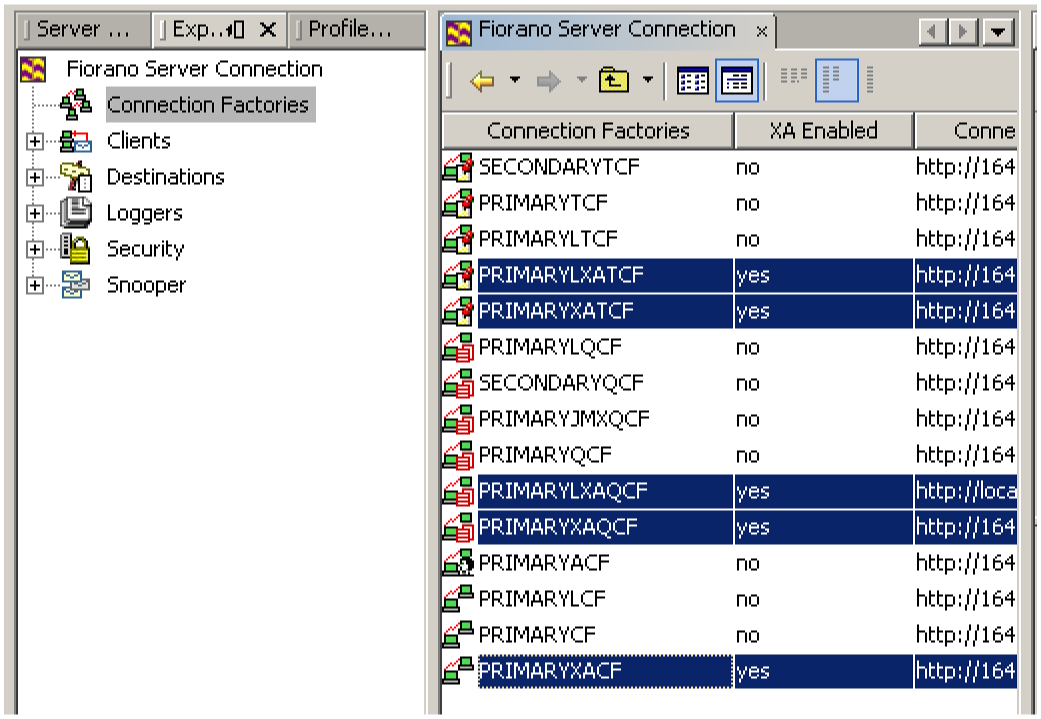
XA requires RDBMS based storage, and can be configured as explained in the 7th point in *Section 23.1 - How to Enable XA.*

23.3 XA Enabled Admin Objects

The FioranoMQ server creates some XA admin objects in addition to those created in the default profile.

23.3.1 Default Admin Objects

Below are admin objects created when XA is enabled in the server.



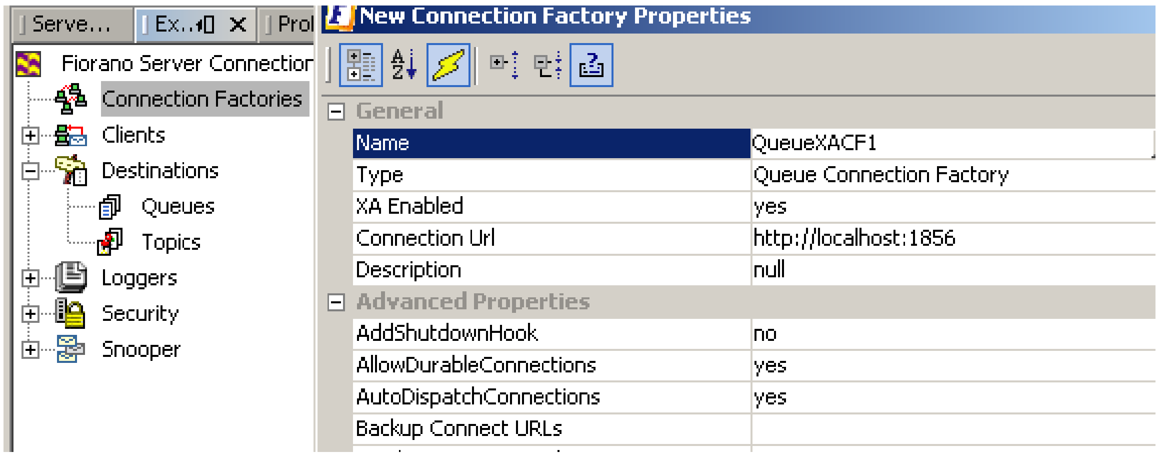
Connection Factories	XA Enabled	Conne
SECONDARYTCF	no	http://164
PRIMARYTCF	no	http://164
PRIMARYLTCF	no	http://164
PRIMARYLXATCF	yes	http://164
PRIMARYXATCF	yes	http://164
PRIMARYLQCF	no	http://164
SECONDARYQCF	no	http://164
PRIMARYJMXQCF	no	http://164
PRIMARYQCF	no	http://164
PRIMARYLXAQCF	yes	http://loca
PRIMARYXAQCF	yes	http://164
PRIMARYACF	no	http://164
PRIMARYLCF	no	http://164
PRIMARYCF	no	http://164
PRIMARYXACF	yes	http://164

Besides the XA enabled connection factory, an RDBMS based default topic and a queue are also created.

23.3.2 Creating XA Enabled Connection Factories

The user can create XA enabled connection factories by connecting Fiorano Studio to the FioranoMQ server. Perform the following steps:

1. Launch Fiorano Studio and connect to the MQ server using the Fiorano Server connection.
2. Right-click on the **Connection Factories** node and select the **Add Connection Factory** option. The **New Connection Factory Properties** dialog box is displayed.
3. Provide information about parameters such as connection URL, XA Enabled, and so on in the **New Connection Factory Properties** dialog box.



General

Name	QueueXACF1
Type	Queue Connection Factory
XA Enabled	yes
Connection Url	http://localhost:1856
Description	null

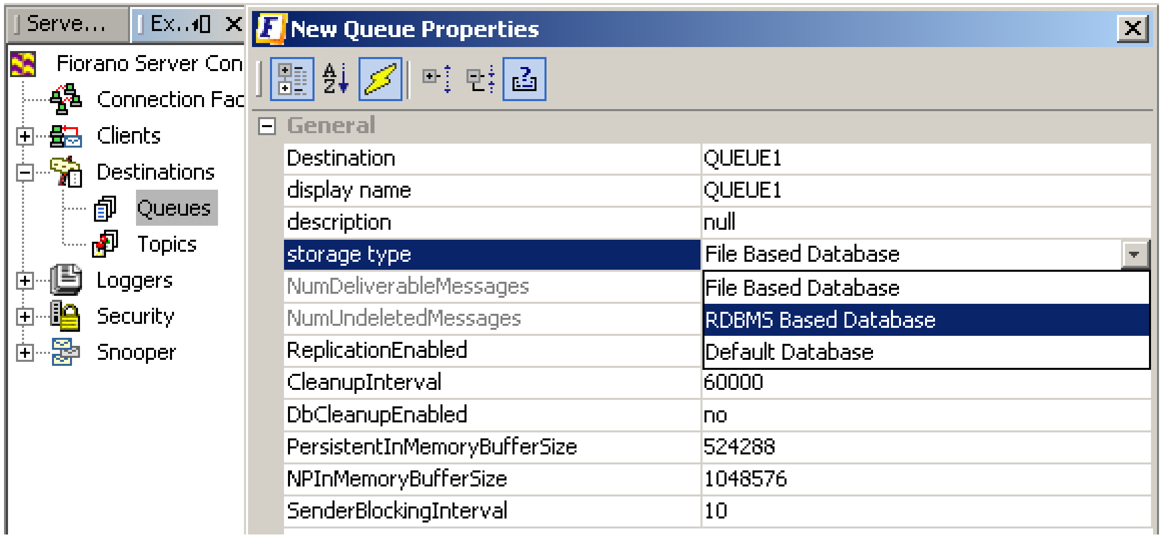
Advanced Properties

AddShutdownHook	no
AllowDurableConnections	yes
AutoDispatchConnections	yes
Backup Connect URLs	

23.3.3 Creating RDBMS Enabled Destinations

As explained earlier XA requires RDBMS based message storage to be enabled. The steps below explain the procedure for creating an RDBMS enabled destination:

1. Launch Fiorano Studio and connect to the MQ server using the Fiorano Server connection.
2. Navigate to **Destinations -> Queues/Topics** and click on **Add Queue/Topic**. The **New Queue Properties** dialog box appears.
3. Edit the properties and specify the storage type as **RDBMS Based Database** in the **New Queue Properties** dialog box.



Destination	QUEUE1
display name	QUEUE1
description	null
storage type	File Based Database
NumDeliverableMessages	File Based Database
NumUndeletedMessages	RDBMS Based Database
ReplicationEnabled	Default Database
CleanupInterval	60000
DbCleanupEnabled	no
PersistentInMemoryBufferSize	524288
NPInMemoryBufferSize	1048576
SenderBlockingInterval	10

Note: For details on how to create XA connection factories using Admin APIs, please refer to 23.9 Admin APIs.

23.4 Usage Scenarios of XA Transactions

23.4.1 As a Standalone Application

A standalone JMS application can use FioranoMQ's implementation of JMS XA API (JMS XA SPI) to participate in a distributed transaction. The JMS application should write the XA specific code to run a XA transaction.

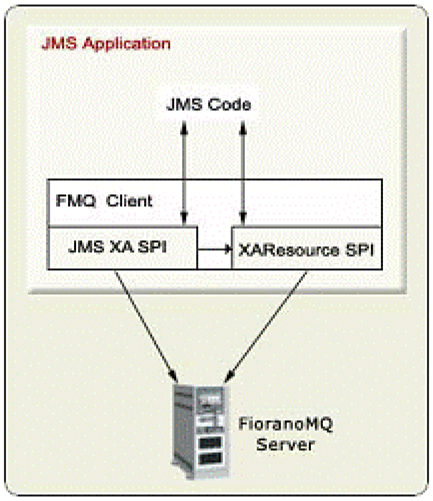


Figure: A stand-alone application uses FioranoMQ XA Support

As shown in the diagram above, the JMS application should:

- Use JMS XA interfaces (XAConnectionFactory, XAConnection, XASession) instead of non-XA interfaces (ConnectionFactory, Connection, Session).
- Use the XAResource interface (XAResource.start() and XAResource.end()) to demarcate transaction boundaries.
- Explicitly commit/rollback the transaction (XAResource.commit() or XAResource.rollback()) as per the business logic.
- Examine the changes that need to be made to the JMS Client application that participates in a distributed transaction.

Follow the steps describe below:

1. Lookup ConnectionFactories and create related JMS resources.

   As the implementation stack for XA-related resources is distinct from non-XA resources, XA connectionFactories need to be created. The Admin API sections of this document explain the creation of ConnectionFactories.

   XAQueueConnectionFactory m_xaQueueConnectionFactory =

   (XAQueueConnectionFactory) ic.lookup(“primaryXAQCF”);

   > The lookup call for these connectionFactories returns an instance of the XAConnectionFactory.

   XAQueueConnection m_xaQueueConnection =

   m_xaQueueConnectionFactory.createXAQueueConnection();

   > Customize all the other JMS-specific resources to either send or receive data. Note the use of the createXAQueueConnection method. The above JMS API call results in the creation of a XAQueueConnection with the MQServer.

   XAQueueSession m_xaQueueSession = m_xaQueueConnection.createXAQueueSession();

   > Create an instance of XAQueueSession. All distributed transactions are associated with a session context. Any operation performed on a session can potentially take part in a distributed transaction.

2. Using XA resources and creating XID.

   The next critical step involves retrieving the XAResource that identify the session context. A uniqueID (XID) is associated with every XAResource to uniquely identify each instance of a distributed transaction. The XID is used to recover failed transactions as well.

   XAResource m_xaResource = m_xaQueueSession.getXAResource();

   String branchId = "BranchID11";

   String globalId = “globalId1”;

   Xid m_xid = new FioranoXid(globalId,branchId.getBytes(),0);

3. Marking the Start and the end transactions.

   Mark the beginning and end of transactions. All operations performed on the session between the start and end statements can be part of distributed transactions.

   ```
   m_xaResource.start(m_xid,XAResource.TMNOFLAGS);

   // Perform all operations on the associated Session

   m_xaResource.end(m_xid,XAResource.TMSUCCESS);
   ```

4. Completing the transaction

   ```
   int type0= m_xaResource.prepare(m_xid);

   m_xaResource.commit(m_xid,false);
   ```

   These APIs are typically used by the Transaction Co-ordinator that manages the transaction across XA Resources. The commit or rollback call completes the transaction. Where applications control distributed transactions, the APIs are used directly by the applications.

   Users are strongly urged to use the JTA specification and read up literature on distributed transactions in order to understand the implementation and usage of the JTA APIs. This section only illustrates the use of JMS APIs for a client application that needs to use distributed transactions. Please view the distributed transaction samples, which are available with the installation package, for more details.

   Normally, a stand-alone application takes part in a distributed transaction when a single server is involved in the transaction. Where more than one server (or different parties) are involved in the distributed transaction, the application must use a third-party external Transaction Manager to co-ordinate the transaction.

23.5 Using FioranoXA with a Pluggable Transaction Manager

In a true distributed transaction, one needs a manager that manages the state of the transaction as it proceeds. With FioranoMQ, any transaction manager that implements the JTA specifications can be used.

- A JMS application uses a Transaction Manager to manage the resources taking part in the transaction.The steps listed below need to be followed to execute a transaction:
  - Acquire the resource providers that take part in the XA transaction. The JMS application gets the resource objects and the XAResource object that takes part in the transaction. For FioranoMQ, it can get the resource object from the XASession object. (**XAConnectionFactory->XAConnection->XASession->XAResource**)
  - Acquire a new transaction object from the Transaction Manager.
  - Enlist the resources in the transaction object. The Transaction Manager internally starts the resources to mark the beginning of the distributed transaction.

- o Perform zero or more operations with the various resources. The operations performed constitute the work done in the transaction.
- o De-list the resources to mark the end of the transaction.
- o Commits/Rollbacks the transaction.

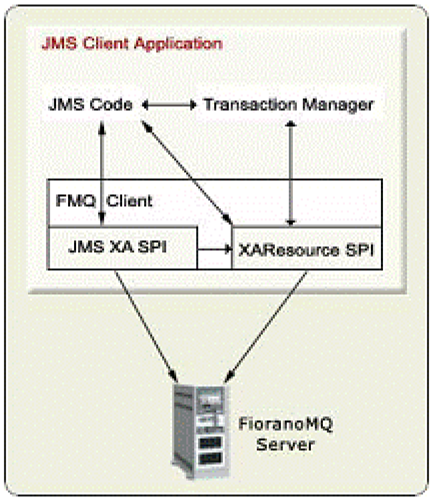


Figure: FioranoMQ XA SPI with an external Transaction Manager FioranoMQ XA support can be used with all available JTA compliant transaction managers. FioranoMQ XA has been successfully integrated with various Transaction Managers by Oracle, oc4j, Orion, and Borland (Visitrans Transaction Manager).

In case you are using a different transaction manager, please contact support@fiorano.com for step-by-step integration instructions.

The following sections explain how to use Orion Transaction Manager and Borland Transaction Manager to manage FioranoXA transactions.

23.6 Using Fiorano XA with the Oc4j Transaction Manager

Using Fiorano XA with the Oc4j Transaction Manager is explained in this section with the help of an example. In this example, a session bean is used with oc4j transaction manager and the oracle 9i application server. The bean client invokes the transact() method of the bean being executed in the oc4j application server.

The work done in this distributed transaction includes:

- Messages are pushed on a Queue residing on the FioranoMQ server.
- Messages are inserted into a table in the Cloudscape database.
- The tasks above are performed in the same distributed transaction.
- The transaction manager manages the transaction and ensures that both the processes are completed successfully.
- Clicking the link and unzipping the file *TestBean.zip*. This zip file contains all the class files of the samples discussed in this guide.

To run the above bean sample with the Orion application server, follow the instructions given below:

For more information on how to configure FioranoMQ with the cloudscape database, please refer to Chapter 6: Configuring Message Store.

1. Install the oracle application server:
2. Install JDeveloper or the oc4j Application server.
3. Configure the Cloudscape database. Configure the orion application server
4. Replace server.xml in *%JDEVELOPER%\j2ee\home\config* directory with *%UNZIP_DIR%\server.xml*. If the server.xml file has applications deployed, add the tag:

   <application name="FMQ samples" path="../demo/fmq"/>
5. Copy the cloudscape libraries, **cloudscape.jar** and **cloudview40.jar** in *%JDEVELOPER%\j2ee\home\lib* directory.
6. Copy classes12.zip in %JDEVELOPER%\j2ee\home\lib and %JDEVEL-OPER%\j2ee\home directory.
7. Add the files <Fiorano_install_dir>\fmq\lib\client\all\\fmq-client.jar to the **oc4j.jar** file, found in %JDEVELOPER%\j2ee\home directory.

   On a Windows platform, this is usually equivalent to \Program Files\Fiorano\FMQ. Here the JDEVELOPER variable represents the base directory of the Orion Application server.
8. Extract the Bean Sample. Unzip the file **fmq.zip**, found in *%UNZIP_DIR%* where the file **TestBean.zip** in the /demo directory of *OC4J_HOME* directory is to be unzipped as well. This creates a folder *fmq* in /demo dir.
9. Deploy the Bean Sample. Restart OC4J using java -jar oc4j.jar. OC4J deploys the new application called FioranoMQ Samples and the bean TestJmsXa with it. The following output is observed if the bean is deployed successfully:

   ```
   Auto-deploying FMQ samples (Assembly had been updated)...
   Auto-deploying test (Class 'TestJmsXaRemote' had been updated)... done.
   ```
10. Run the Bean Sample. Run the EJB client using run-client TestJmsXaClient. (This client can be found in jdeveloper\j2ee\home\demo\fmq\test.) Ensure that theoc4j.jar, jass.jar and ejb.jar files are in the classpath before running the client.

Examine the details of the bean as well as its client to see how the distributed transaction involving the transaction manager work.

1. Lookup of the Home Interface and invoking methods of EJB.

   The reference to the home interface of the deployed bean can be accessed through the JNDI lookup calls. The lookup of the bean returns the home interface of the deployed bean. A remote interface is created from the home interface.

   ```
   homeInterface = (TestJmsXaHome) ctx.lookup("java:comp/env/TestJmsXa");

   remoteInterface     = homeInterface.create();
   ```

   The various methods of the bean can be invoked through its remote interface. The transact method, which performs all the steps required for running the distributed transaction in the test bean, can be invoked using:

   ```
   remoteInterface.transact();
   ```

2. Transaction Reference

   In implementating the transact method, reference to the transaction object can be obtained through the bean context. This transaction object is used to define the transaction boundaries.

   ```
   transactionManager  =  (TransactionManager) ctx.getUserTransaction();

   transactionManager.begin();

   transaction = transactionManager.getTransaction();
   ```

3. Enlisting a Transaction

   Resources should be enlisted within the transaction to inform the transaction about their participation. Reference to the resources can be acquired through the application server or directly from the resource provider. Please check the section '*To run as Standalone application'* to obtain the XA resource from the resource provider.

   ```
   transaction.enlistResource(xaresource);
   ```

4. Begin Transaction

   Begin the transaction after enlisting the participating resources. All the work done after starting the transaction is associated with the transaction. One example of such work could be publishing a message on a queue.

   ```
   transaction.begin();
   ```

5. Delisting Transaction

   After performing the task in the transaction, the resources should be removed from the transaction. Delisting the resources informs the transaction that any work performed after delisting should not be made part of the transaction.

   ```
   {

   transaction.delistResource(xaresource,XAResource.TMSUCCESS);
   ```

   The parameter, XAResource.SUCCESS indicates that the transaction has ended successfully. If an exception occurs while a task is being performed, the application can remove the resource with XAResource.FAIL.

6. Committing Transaction

   Committing a transaction commits the work performed by all the participating resources. The transaction is committed by a two-phase commit protocol:

   ```
   {

   transactionn.commit() ;
   ```

23.7 Using FioranoXA with the Borland Transaction Manager

FioranoMQ can be used to run the distributed transaction in which the Borland Transaction manager is the co-ordinator. Consider a scenario where an enterprise application takes part in a distributed transaction in which it receives a message from a JMS queue and inserts that message in the cloudscape database. The implementation of this scenario can be downloaded from the fiorano_borland.zip link.

23.7.1 Sample Details

Unzip the file sample downloaded from *fiorano_borland.zip*. This file contains all the necessary files required to run the sample.

CreateCloudScapeDB.java - This file creates the database on the cloudscape database.

TestXaCloudScapeSampleWithTM.java - This file runs the distributed transaction in which messages received from a queue are inserted into the cloudscape database tables.

23.7.2 Integrate Fiorano with Borland Enterprise server

Integrate the Borland Application server with FioranoMQ using the instructions present in Chapter 27 Application Server Integration.

23.7.3 Configure Classpath

1. Configure the classpath by adding the following Borland related files to the run-client.bat (run-client.sh on UNIX Systems) file, which is available in the fmq/bin/ directory of the FioranoMQ installation package. The files *asrt.jar, vbjorb.jar, lm.jar, vbsec.jar, xmlrt.jar, jsse.jar, jaas.jar, jcert.jar, jnet.jar, vbejb.jar* are available in the library directory of the Borland installation. The sample run-client.bat (run-client.sh) can be downloaded as well.
2. Modify the FMQ_DIR variable and BORLAND_DIR variable within the script before using it. Add the cloudscape related files to the classpath.

   cloudview40.jar, cloudscape.jar.

3. Modify the CLOUDSCAPE_DIR in the downloaded script file.

Database Creation:

Create two instances of cloudscape databases using run-client COM.CloudScape.tools.cview

Create tables in the database using the sample CreateCloudscapeDB available from the downloaded zip file.

Usage:

run-client CreateCloudScapeDB <databasepath>

<databasepath> represents the path of the cloudscape database.

Run the distributed transaction:

Run the distributed transaction using the sample TestXaCloudScapeSample with TM available in the downloaded zip file.

Usage:

run-client TestXaCloudScapeSampleWithTM [host-no] [port-no] [num-of-msgs] [xaqcf-name] [queue-name]

Given below is a description of the command line parameters:

<host-no> This parameter specifies the machine name or IP address on which the FioranoMQ server is running.

<port-no> This parameter specifies the port number of the machine on which the FioranoMQ server is running.

<num-of-messages> This parameter specifies the number of messages that should be received from the queue and inserted into the cloudscape database in a distributed transaction.

<xaqcf-name> This parameter specifies the name of the XA enabled queue connection factory.

<queue-name> This parameter specifies the name of the queue from which messages should start.

Example:

run-client TestXaCloudScapeSampleWithTM localhost 1856 10 primaryXAQCF primaryRDBMSQueue

Sample Details

Details of the sample are examined to see how the distributed transaction works with the Borland transaction manager.

- Transaction Reference

  This is the reference to the transaction object. This reference is provided to the application by the transaction co-ordinator. Here, the transaction co-ordinator is the Borland Transaction Manager, which provides the application with a transaction object.

  ```
  transactionManager  =  TransactionManager.getTransactionManagerImpl();

  transactionManager.begin();

  transaction = transactionManager.getTransaction();
  ```

- Enlisting Transaction

  The resources should be enlisted within the transaction to inform the transaction about their participation. Reference to the resources is provided by the resource provider.

  ```
  transaction.enlistResource(xaresource);
  ```

- Begin Transaction

  Begin the transaction after enlisting the participating resources. All the work done after starting the transaction is associated with the transaction. An example of work done is the publishing of a message on a queue.

  ```
  transaction.begin();
  ```

- Delisting Transaction

  After performing the task allocated within the transaction, the resources should be removed from the transaction. Removing the resources inform the transaction indicates that any work performed after this transaction should not be made part of the transaction.

  ```
  {

  transactionn.delistResource(xaresource,XAResource.TMSUCCESS);
  ```

  The parameter, XAResource. **SUCCESS** indicates that the transaction has ended successfully. If an exception occurs while performing a task, the application can remove the resource with XAResource.FAIL.

- Committing Transaction

  Committing a transaction commits the work performed by all the participating resources. Commit is performed by a two-phase commit protocol.

  ```
  {

  transactionn.commit() ;
  ```

23.8 Transactions with J2EE

The following section provides the information required to integrate and run UserTransactions using Oracle 9i and Weblogic 7.0 Application servers.

23.8.1 Integrating with WebLogic 7.0 Application Server

1. Execute the following steps to integrate Fiorano's XAResource with the Weblogic 7.0 Application server:
2. Extract the files present in weblogic_fiorano.zip to a separate folder (UNZIP_DIR). This folder contains all the files required to perform the steps listed below:
3. Install Weblogic Platform 7.0.1.0
4. Copy "%FMQ_DIR%\lib\client\all\fmq-client.jar" to "%WL_HOME%\server\lib".
5. Include the file, %UNZIP_DIR%\fiorano.zip,%WL_HOME%\server\lib\fmq-client.jar in the classpath of %WL_HOME%\samples\server\config\examples\setExamplesEnv.cmd and %WL_HOME%\server\bin\startWLS. This fiorano.zip file contains the wrapper classes used for the interception of calls made by the application server.
6. Navigate to the folder %WL_HOME%\samples\server\src\examples\ejb20\message and copy the files *MessageTraderBean.java*, *Client.java, ejb-jar.xml* and *weblogic-ejb-jar.xml* extracted from the file *WeblogicIntegration.zip*, which can be found in %UNZIP_DIR%.
7. Run %WL_HOME%\samples\server\config\examples\setExamplesEnv.cmd on the DOS prompt. This provides the environment required for running the sample.
8. Navigate to the folder %WL_HOME%\samples\server\src\examples\ejb20\message using the aforementioned DOS prompt and execute the file *build.cmd* present in the zip file *WeblogicIntegration.zip*, which can be found in %UNZIP_DIR%. This compiles the file *Client.java* and adds it to the folder %WL_HOME%\samples\server\stage\examples\clientclasses\ examples\ejb20\message since the package of *Client.java* is *examples.ejb23.message*. In addition this rebuilds *ejb20_message.jar* and adds it to the folder %WL_HOME%\samples\server\config\examples\applications.
9. Include the FioranoWeblogicMapper in the weblogic startup.

   The FioranoWeblogicMapper class maps the JNDI objects in the FioranoMQ name space (XA enabled connectionfactories, XA enabled Queues, and XA enabled Topics) and binds them to objects in the Weblogic JNDI. These objects can be looked up using Weblogic's InitialContext in the following manner:

   Ictx.lookup("fiorano_wl_PRIMARYXAQCF") etc..

   The FioranoWeblogicMapper maps the FioranoMQ administered objects to the Weblogic Naming tree in a manner that allows *primaryXAQCF* in FioranoMQ's JNDI to be mapped onto *fiorano_wl_PrimaryXAQCF* in Weblogic's JNDI.

10. Copy the attached FioranoWeblogicMapper.zip to the classpath set in the startExamplesServer (or startWLS.cmd) file.
11. Open the Weblogic adminConsole. Select deployments\startup and shutdown.
12. Select **configure a new Startup class**. Under the configuration tab, specify the following:
    a. Name FioranoWeblogicMapper
    b. ClassName FioranoWeblogicMapper
    c. Deployment order 0
    d. Arguments None
13. Start the FioranoMQ server (with XA enabled). Navigate to the folder %WL_HOME%\server\bin and start the Oracle Weblogic Application server by executing the batch file:

    startWLS.cmd

The bean is deployed on the Oracle Weblogic Application server. It is a message driven bean listening on primaryTopic.

14. Use a normal publisher (run-client Publisher) to publish a message on a primaryTopic. The onMessage of the bean is invoked, which in turn calls the transact() method of the MDB.

    transact() method performs the following steps:
    a. Looks up XA Queue Connection Factory.
    b. Starts the usertransaction.
    c. Sends a message on the primaryRDBMSQueue.
    d. Commits the userTransaction.

Note: A receiver on primaryRDBMSTopic receives the message only when the usertransaction is committed. (Similar tests can be performed for ut.rollback() and other methods.)

15. Add fiorano.zip to the classpath of build.cmd.
16. To compile the samples, execute: run build.cmd

23.8.2 Integrating with Oracle 9i Application Server

Perform the following steps to integrate Fiorano's XAResource with the Oracle 9i Application server:

1. Unzip the *orion_fiorano.zip* file, which contains the files required to run the sample discussed in this section.
2. Unzip and add classes to fmq-client.jar found in the %FMQ_HOME%\fmq\lib\client\all directory to the ORACLE_HOME\j2ee\home\oc4j.jar file.
3. Replace *server.xml* located in OC4J_HOME\config with the one found *in orion_fiorano.zip* file. If your *server.xml* file has any applications deployed add the tag below to the application-server tag of *server.xml*, instead of replacing the file.

   <application name="FMQ samples" path="../demo/fmq"/>
4. Unzip the file *fmq_bean.zip* (also found in the *orion_fiorano.zip* file mentioned above), located in the /demo directory of OC4J_HOME directory. This creates a folder called fmq in the /demo dir with the java and class files *TestJmsXaClient*, *TestJmsXa.java, TestJmsXaRemote.java*, and *TestJmsXaHome*.
5. Change the ORACLE database ConnectionParameters (URL, username, password) in the file *TestJmsXa.java* in keeping with the Oracle installation and recompile the file. The files *ejb.jar, jaas.jar,* and *oc4j.jar* must be in the classpath when compiling and running the files.
6. Replace the file CMTQueueSession.class with the attached CMTQueueSession.class in the oc4j.jar file. Extract the CMTQueueSession.class file from orion_fiorano.zip file. Make sure that the package structure of this file remains intact while replacing the class in the oc4j.jar file. (The package is com.evermind.server.jms.CMTQueueSession.class).
7. Start OC4J. OC4J deploys the application called FioranoMQ Samples and the bean TestJmsXa with it. If the bean is deployed successfully the output should be:

   Auto-deploying FMQ samples (Assembly had been updated)...

   Auto-deploying test (Class 'TestJmsXaRemote' had been updated)...
8. Start the FioranoMQ server and a JMS QueueReceiver that is receiving messages on primary RDBMSQueue.
9. Run the client to the bean, TestJmsXaClient (run_oracle TestJmsXaClient). In the client, the map() method of the bean is called and this maps the FioranoMQ XA enabled connectionfactories to the oracle9i JNDI.
10. The client invokes the transact method in which a UserTransaction object is received. The UserTransaction is started, after which a message is sent to the FioranoMQ primaryRDBMSQueue. Following this message the user transaction is committed. The QueueReceiver receives the message only when the usertransaction is committed.

23.9 Admin APIs

FioranoMQ provides a comprehensive set of administration APIs that allow the enterprise administrator to manage a number of facilities such as Distributed Transactions, Topics, Queues, and XA ConnectionFactories.

All Admin requests are serviced by the MQ Server listening on a dedicated Admin port (default is 1856). The admin port can be configured through *ConnectionManager.xml*.

All Administered Object names are case-insensitive. Complete samples that illustrate the use of Admin APIs to create, manage and delete XA resources are available in the /fmq/samples/DistributedTransactions directory within the FioranoMQ installation package.

XA Connection Factories

XAConnectionFactory object is a JTS support supplied by every JMS provider to create XAConnection objects.

XAConnectionFactory objects are JMS administered objects similar to ConnectionFactory objects. They can be looked up using the Java Naming and Directory Interface (JNDI) API.

23.9.1 Creating XA Connection Factories

XA connection factories can be created using FioranoMQ's Admin API. The procedure for creating a XA connection factory is given below:

1. Look up an admin connection factory. For example, look up the **primaryACF** thatis created by default.
2. Create an admin connection using the admin connection factory.
3. Get the admin service from the admin connection.
4. Create the XA Connection factory metadata. The possible options are XAQueueConnectionFactoryMetaData, XATopicConnectionFactoryMetaData and UnifiedXAConnectionFactoryMetaData.
5. Set the name of the factory.
6. Set a brief description of the factory. (This step is optional.)
7. Set the connect URL of this factory. Connect URL is the URL of the server to which a connection is created.
8. Set the semi colon separated backup URLs. Backup URLs are the URLs to which a connection is created if the connect URL is down.

Note: For details, refer to the section, 29.2 N Failover URL Support.

23.9.2 Creating a XA Queue Connection Factory

This section explains the creation of XAQueue Connection factories:

1. Creating the Admin Service

   MQ Admin Service creates new connection factories on the FioranoMQ server. The instance of admin service can be obtained from the admin connection.

   acf = (MQAdminConnectionFactory) ic.lookup ("primaryACF");

   ac = acf.createMQAdminConnection ("admin", "passwd");

   adminService = ac.getMQAdminService();

2. XAQueueConnectionFactory Metadata

   Create an XAQueueConnection factory metadata, specifying the name, description and the primary connection URL.

   xametaData = new XAQueueConnectionFactoryMetaData();

   xametaData.setName(“myXAQCF”);

   xametaData.setDescription ("XA Queue Connection Factory");

   xametaData.setConnectURL ("http://localhost:1856");

3. Create XAQueueConnectionFactory

   Create the XAQueue connection factory using the adminService and the connection factory metadata.

   adminService.createXAQueueConnectionFactory (*xametaData*);

23.9.3 Creating a XA Topic Connection Factory

This section explains the creation of XATopic connection factories:

1. Creating the Admin Service

   MQ Admin Service creates new connection factories on the FioranoMQ server. The instance of admin service can be obtained from the admin connection.

   acf = (MQAdminConnectionFactory) ic.lookup ("primaryACF");

   ac = acf.createMQAdminConnection ("admin", "passwd");

   adminService = ac.getMQAdminService();

2. XATopicConnectionFactory Metadata

   Create a XATopicConnection factory metadata, specifying the name, description, primary connection URL.

   xametaData = new XATopicConnectionFactoryMetaData();

   xametaData.setName (“myXATCF”);

xametaData.setDescription ("XA Topic Connection Factory");

xametaData.setConnectURL ("http://localhost:1856");

3. Create XATopicConnectionFactory

   Create the XATopicConnection factory using the adminService and the connection factory metadata.

   adminService.createXATopicConnectionFactory (*xametaData*);

23.9.4 Creating a Unified XA Connection Factory

This section explains the creation of the Unified XA connection factories:

1. Creation of Admin Service

   MQ Admin Service creates new connection factories on the FioranoMQ server. The instance of admin service can be obtained from the admin connection.

   acf = (MQAdminConnectionFactory) ic.lookup ("primaryACF");

   ac = acf.createMQAdminConnection ("admin", "passwd");

   adminService = ac.getMQAdminService();

2. UnifiedXAConnectionFactory Metadata

   Create a UnifiedXAConnection factory metadata, specifying the name, description and the primary connection URL.

   xametadata = new UnifiedXAConnectionFactoryMetaData();

   xametadata.setName("myXAcf");

   xametadata.setDescription ("XA Unified Connection Factory");

   xametadata.setConnectURL ("http://localhost:1856");

3. Create UnifiedXAConnectionFactory

   Create the UnifiedXaConnection factory using the adminService and the connection factory metadata.

   adminService.createXAConnectionFactory (xametadata);

23.9.5 Deleting XA Connection Factories

The procedure for deleting a XA connection factory is:

1. Look up an admin connection factory. For example, look up the **primaryACF** that is created by default.
2. Create an admin connection using the admin connection factory.
3. Get the adminservice from the admin connection.
4. Delete the XA connection factory using the admin service.

23.9.6 Deleting a XA Queue Connection Factory

This section explains the deletion of the XA Queue connection factories:

1. Creating the Admin Service

   MQ admin service deletes new connection factories on the FioranoMQ server. The instance of admin service can be obtained from the admin connection.

   ```
   acf = (MQAdminConnectionFactory) ic.lookup ("primaryACF");
   ```

   ```
   ac = acf.createMQAdminConnection ("admin", "passwd");
   ```

   ```
   adminService = ac.getMQAdminService();
   ```

2. Delete XAQueue Connection Factory

   Delete the XAQueue connection factory using the admin service and the connection factory name.

   ```
   adminService.deleteXAQueueConnectionFactory (“myxaqcf”);
   ```

23.9.7 Deleting a XA Topic Connection Factory

This section explains the deletion of the XA Topic connection factories:

1. Creating the Admin Service

   MQ admin service deletes new connection factories on the FioranoMQ server. The instance of admin service can be obtained from the admin connection.

   ```
   acf = (MQAdminConnectionFactory) ic.lookup ("primaryACF");
   ```

   ```
   ac = acf.createMQAdminConnection ("admin", "passwd");
   ```

   ```
   adminService = ac.getMQAdminService();
   ```

2. Delete the XA Topic Connection Factory

   Delete the XATopic connection factory using the admin service and the connection factory name.

   ```
   adminService.deleteXATopicConnectionFactory (“myxatcf”);
   ```

23.9.8 Deleting a Unified XA Connection Factory

This section explains how to delete the Unified XA connection factories:

1. Creating the Admin Service

   MQ admin service deletes new connection factories on the FioranoMQ server. The instance of admin service can be obtained from the admin connection.

   ```
   acf = (MQAdminConnectionFactory) ic.lookup ("primaryACF");
   ```

   ```
   ac = acf.createMQAdminConnection ("admin", "passwd");
   ```

   ```
   adminService = ac.getMQAdminService();
   ```

2. Delete UnifiedXA Connection Factory

   Delete the unified XA connection factory using the admin service and the connection factory name.

   ```
   adminService.deleteXAConnectionFactory (“myxacf”);
   ```

   A destination object is a JMS administered object containing configuration information that is created by an administrator and later used by JMS clients. JMS clients can find destinations by looking them up in a JNDI namespace.

23.10 Destinations

In FioranoMQ 6.1 beta upwards, *storage type* has been introduced in the destinations. *Storage types* define the type of storage that should be used to store destinations. The possible storage types are file-based and RDBMS based stores. By default, the storage type is file based.

```
// For File Based destination

metadata1.setStorageType(IFioranoConstants.FILE_BASED_DATABASE);

// For RDBMS based destination

metadata1.setStorageType(IFioranoConstants.RDBMS_BASED_DATABASE);
```

If a destination is file based, then all the persistent messages that are sent on this destination are stored in files. If a destination is RDBMS based, then all the persistent messages are stored in RDBMS.

Storage type is specified at the time of creating a destination.

Below are the steps to create a destination using *storage type*:

1. Look up an admin connection factory.

   Create an Admin connection.

2. Obtain the Admin Service from the Admin connection.
3. Create a new metadata. It can be either QueueMetaData or TopicMetaData.
4. Set the name of the destination.
5. Set the storage type of the destination.
6. Create the destination using the admin service.

23.11 Queues

This section explains the creation of Queue objects:

1. Creating the Admin Service

   MQ admin service creates queues on the FioranoMQ server. The instance of admin service can be obtained from the admin connection.

   acf = (MQAdminConnectionFactory) ic.lookup ("primaryACF");

   ac = acf.createMQAdminConnection ("admin", "passwd");

   adminService = ac.getMQAdminService();

2. Queue Metadata

   Create a queue metadata, specifying the name, description and the storage type. The storage type of the destination can be RDBMS_BASED_DATABASE or FILE_BASED_DATABASE

   QueueMetaData metadata = new QueueMetaData ();

   metadata.setName (“myqueue”);

   // For RDBMS based storage type

   metadata.setStorageType(IFioranoConstants.RDBMS_BASED_DATABASE);

   or

   // For File based storage type

   metadata.setStorageType(IFioranoConstants.FILE_BASED_DATABASE);

3. Create the Queue

   Create the Queue object using the adminService and the queue metadata.

   adminService.createQueue (metadata);

23.12 Topics

This section explains the creation of Topic objects:

1. Creating the Admin Service

   MQ Admin Service creates topics on the FioranoMQ server. The instance of admin service can be obtained from the admin connection.

   ```
   acf = (MQAdminConnectionFactory) ic.lookup ("primaryACF");

   ac = acf.createMQAdminConnection ("admin", "passwd");

   adminService = ac.getMQAdminService();
   ```

2. Topic Metadata

   Create a Topic metadata, specifying the name, description and the storage type. The storage type of the destination can be RDBMS_BASED_DATABASE or FILE_BASED_DATABASE

   ```
    TopicMetaData metadata = new TopicMetaData ();

    metadata.setName (“mytopic”);

   // For RDBMS based storage type

   metadata.setStorageType(IFioranoConstants.RDBMS_BASED_DATABASE);

   or

   // For file based storage type

   metadata.setStorageType(IFioranoConstants.FILE_BASED_DATABASE);
   ```

3. Create the Topic

   Create the Topic object using the adminService and the topic metadata.

   ```
   adminService.createTopic(metadata);
   ```

23.13 Transactions

The Transaction interface allows operations to be performed against the transaction in the target Transaction object. A Transaction object is created corresponding to each global transaction creation in which one or more resources participate. In FioranoMQ, the resource (javax.transaction.xa.XAResource object) is available which participates in distributed transactions. Refer to JTA specification for more details on XAResource object.

FioranoMQ provides the following admin APIs related to distributed transactions:

Get All Transactions Returns an enumeration of XIDs with the given status.

Get Transaction status Returns the status of the transaction specified with the XID.

23.13.1 Get All Transactions

This function returns the enumeration of all the XIDs with a given status. The status of a transaction can be started, ended, suspended or prepared.

// Status is equal to start

fiorano.jms.services.IFioranoConstants.START;

// Status is equal to end

fiorano.jms.services.IFioranoConstants.END;

// Status is equal to suspend

fiorano.jms.services.IFioranoConstants.SUSPEND;

// Status is equal to prepared

fiorano.jms.services.IFioranoConstants.PREPARED;

Below are the instructions to return all the transactions with a ‘given’ status running on the FioranoMQ server.

1. Look up an admin Connection Factory
2. Create an admin connection
3. Obtain the admin service from the admin connection
4. Retrieve all the transactions with a given status

This section explains how to retrieve the list of transactions that have started :

1. Creating the Admin Service

   MQ Admin Service retrieves the list of transactions from the FioranoMQ server. The instance of admin service can be obtained from the admin connection.

   acf = (MQAdminConnectionFactory) ic.lookup ("primaryACF");

```
ac = acf.createMQAdminConnection ("admin", "passwd");

adminService = ac.getMQAdminService();
```

2. Transaction List

Alist of the transactions, given their status, can be retrieved from the admin service. To retrieve a list of transactions that have started, the code snippet given below is to be used. This returns all transactions currently present in the MQ server started in the given state

```
Enumeration enum = adminService.getAllTransactions(IFioranoConstants.START);
```

3. Retrieve Transactions

   Individual transactions or XIDs can be retrieved from a list of transactions using the code below:

   ```
   while (enum.hasMoreElements())

   {

   Xid xid = (Xid) enum.nextElement();

   globaltrxId = xid.getGlobalTransactionId();

   branchId  = xid.getBranchQualifier();

   formatId = xid.getFormatId();

   }
   ```

   This function returns the status of a transaction, which is specified with a XID. The returned status can be started, ended, suspended or prepared.

23.13.2 Get Transaction Status

```
// Status is equal to start

fiorano.jms.services.IFioranoConstants.START;

// Status is equal to end

fiorano.jms.services.IFioranoConstants.END;

// Status is equal to suspend

fiorano.jms.services.IFioranoConstants.SUSPEND;

// Status is equal to prepared

fiorano.jms.services.IFioranoConstants.PREPARED;
```

Below are the instructions to retrieve the status of a given transaction:

1. Look up an admin connection factory.
2. Create an admin connection.
3. Obtain the admin service from the admin Connection.
4. Retrieve the status of a given transaction.

This section explains how to obtain the status of a given transaction:

1. Creating the Admin Service

   MQ Admin Service obtains the status of a given transaction from the FioranoMQ server. The instance of admin service can be retrieved from the admin connection.

   a. acf = (MQAdminConnectionFactory) ic.lookup ("primaryACF");

   b. ac = acf.createMQAdminConnection ("admin", "passwd");

   c. adminService = ac.getMQAdminService();

2. Retrieve Transaction Status

   The status of a given transaction, specified by XID, can be retrieved using the admin service object.

   byte status = adminService.getTransactionStatus(xid);

   Possible options for the status byte:

   a. IFioranoConstants.START - specifies that the transaction is in the started state.

   b. IFioranoConstants.END - specifies that the transaction is in the ended state.

   c. IFioranoConstants.SUSPEND - specifies that the transaction suspended state.

   d. IFioranoConstants.PREPARED - specifies that the transaction is in the prepared

Chapter 24: JMX Notification

24.1 JMX Notifications generated by the server

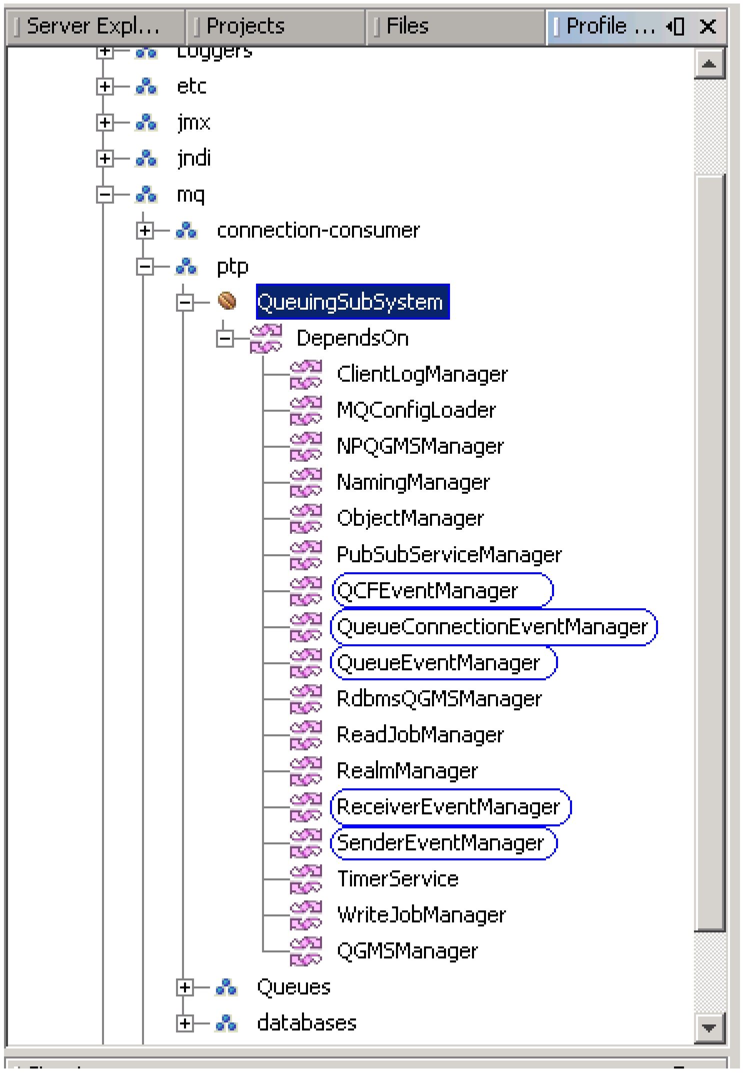
The table below shows a list of Mbeans that generate JMX Notifications along with the conditions in which these notifications are generated.

Notification Condition	**MBean's Object Name**
Pubsub Notifications	
Topic Creation and Deletion	Fiorano.jmx.notifications:ServiceType=EventManager,Name=TopicEventManager
Opening, Closing and Setting of client ID on a Topic Connection	Fiorano.jmx.notifications:ServiceType=EventManager,Name=TopicConnectionEventManager
Creation and Deletion of Topic Connection Factory	Fiorano.jmx.notifications:ServiceType=EventManager,Name=TCFEventManager
Opening and Closing of a Topic Subscriber	Fiorano.jmx.notifications:ServiceType=EventManager,Name=SubscriberEventManager
Opening and Closing of a Topic Publisher	Fiorano.jmx.notifications:ServiceType=EventManager,Name=PublishEventManager
Ptp Notifications	
Opening and Closing of a Queue Sender	Fiorano.jmx.notifications:ServiceType=EventManager,Name=SenderEventManager
Opening and Closing of a Queue Receiver	Fiorano.jmx.notifications:ServiceType=EventManager,Name=ReceiverEventManager
Queue Creation and Deletion	Fiorano.jmx.notifications:ServiceType=EventManager,Name=QueueEventManager
Opening, Closing and setting of client ID on a Queue Connection	Fiorano.jmx.notifications:ServiceType=EventManager,Name=QueueConnectionEventManager
Creation and Deletion of a Queue Connection Factory	Fiorano.jmx.notifications:ServiceType=EventManager,Name=QCFEventManager
Number of messages in a queue exceeding the threshold defined at queue level	Fiorano.jmx.notifications:ServiceType=EventManager,Name=QueueEventManager

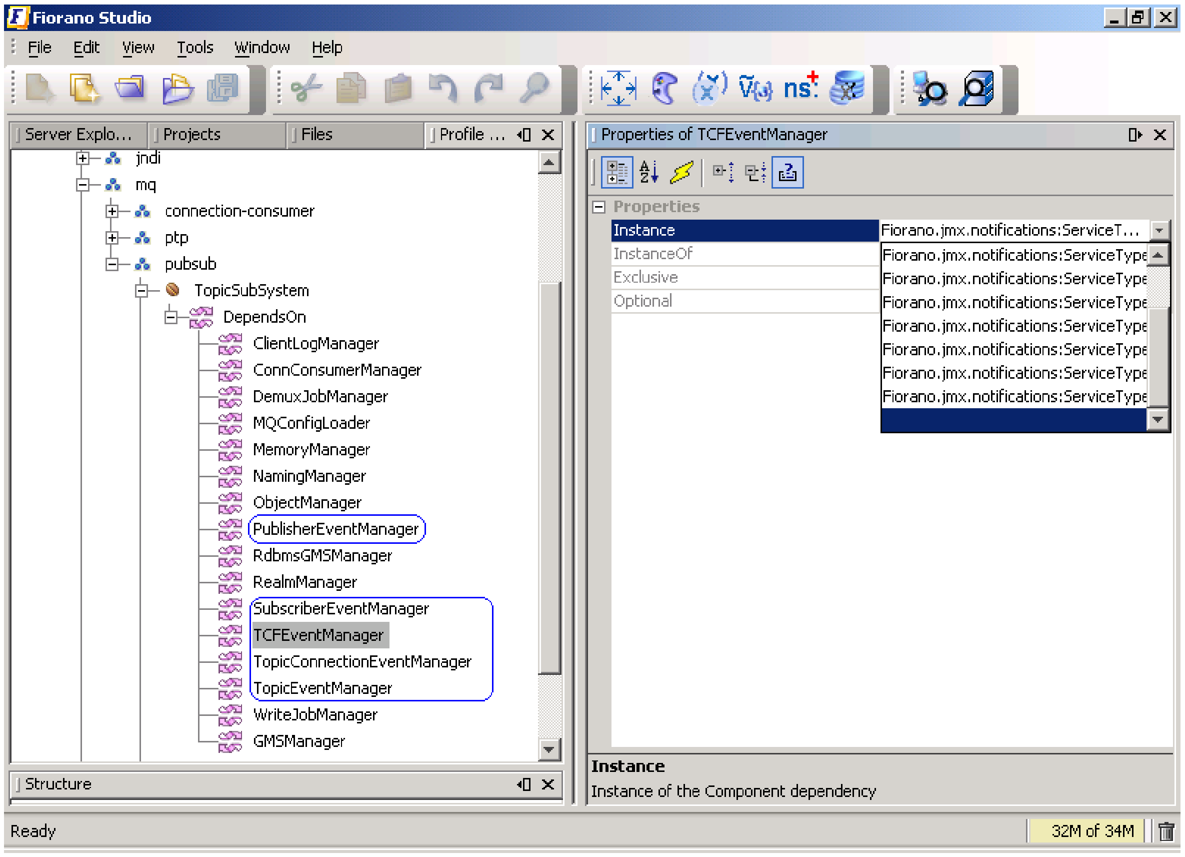
24.2 Enabling/Disabling Notifications

By default all events are enabled. However, it is possible to turn on or off selective events in the server.

The server requires an instance of EventManager (which Is an implementation of NotificationBroadCaster) for each type of event. These event managers are passed to QueuingSubSystem and TopicSubSystem as dependencies. If an Event Manager dependency is left unresolved the event corresponding to that dependency is turned off.



The figure above shows all dependencies of Queuing Sub System in FioranoMQ. To turn off Queue Events set QueueEventManager to null. Similarly, to turn off Topic Connection Events set TCFEventManager to null, as shown in the figure below.



Instance	Fiorano.jmx.notifications:ServiceT...
InstanceOf	Fiorano.jmx.notifications:ServiceType
Exclusive	Fiorano.jmx.notifications:ServiceType
Optional	Fiorano.jmx.notifications:ServiceType
	Fiorano.jmx.notifications:ServiceType
	Fiorano.jmx.notifications:ServiceType
	Fiorano.jmx.notifications:ServiceType
	Fiorano.jmx.notifications:ServiceType

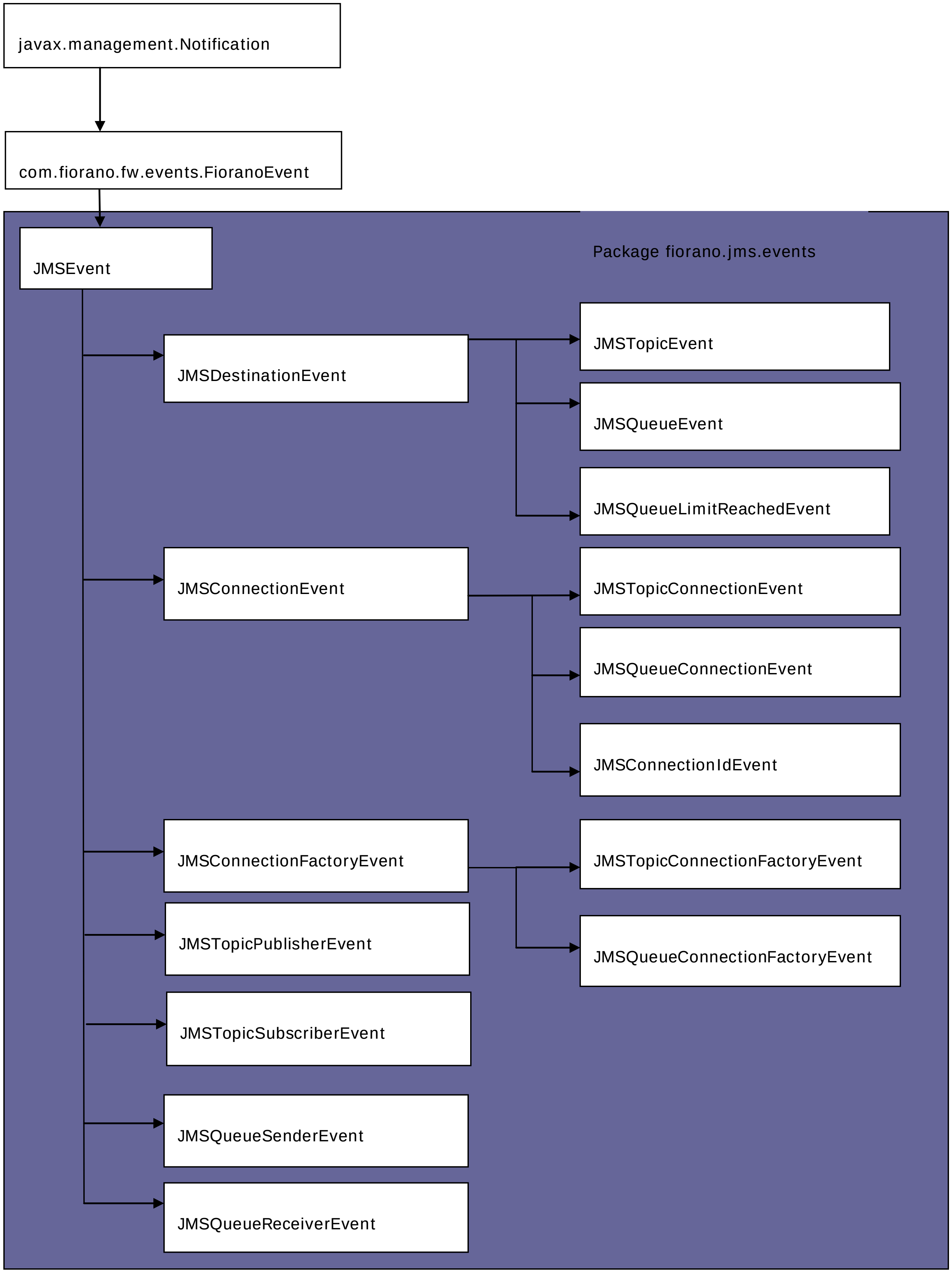
Instance
Instance of the Component dependency

Ready 32M of 34M

24.3 Notification Classes

The FioranoMQ Server, when generating an event, creates an instance of a class depending on the event being generated. The FioranoMQ Server then inserts the required information within this instance raises the event. The diagram below shows various Notification classes and the relationships between them.

The diagram, below, shows the event ID (in blue) set by the server when generating the event. This ID can be used by any application to specify a filter when receiving events.


javax.management.Notification
com.fiorano.fw.events.FioranoEvent
JMSEvent
Package fiorano.jms.events
JMSDestinationEvent
JMSTopicEvent
JMSQueueEvent
JMSQueueLimitReachedEvent
JMSConnectionEvent
JMSTopicConnectionEvent
JMSQueueConnectionEvent
JMSConnectionIdEvent
JMSConnectionFactoryEvent
JMSTopicConnectionFactoryEvent
JMSQueueConnectionFactoryEvent
JMSTopicPublisherEvent
JMSTopicSubscriberEvent
JMSQueueSenderEvent
JMSQueueReceiverEvent

Chapter 25: Online Configuration Through Studio

The Admin Studio allows the user to manage instances of the FioranoMQ Server while it is running. This mode of editing the configuration is called **Online Configuration.** This section of the document provides details of online configuration using the Admin Studio.

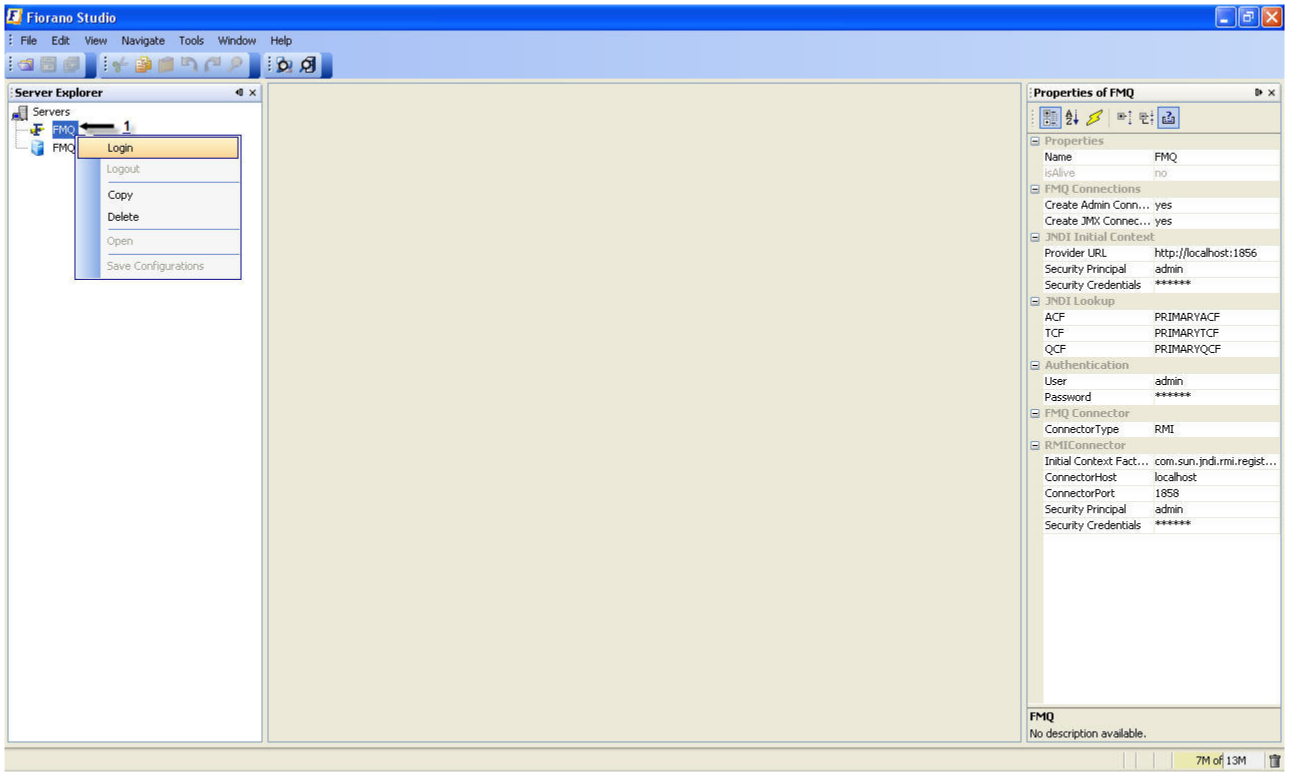
25.1 Connecting to the FioranoMQ Server

25.1.1 Over Default Configuration

The steps to be taken for default configuration are given below:

1. To establish a connection, select the **FMQ** node (marked '1' in the figure below). Specify the **ConnectorPort** and **Provider URL** and or other parameters in the **Properties of FMQ** panel.

2. Right-click **FMQ** and select **Login** from the pop-up menu. This will create a connection to the server.



Properties	
Name	FMQ
isAlive	no
FMQ Connections	
Create Admin Conn...	yes
Create JMX Connec...	yes
JNDI Initial Context	
Provider URL	http://localhost:1856
Security Principal	admin
Security Credentials	******
JNDI Lookup	
ACF	PRIMARYACF
TCF	PRIMARYTCF
QCF	PRIMARYQCF
Authentication	
User	admin
Password	******
FMQ Connector	
ConnectorType	RMI
RMIConnector	
Initial Context Fact...	com.sun.jndi.rmi.regist...
ConnectorHost	localhost
ConnectorPort	1858
Security Principal	admin
Security Credentials	******

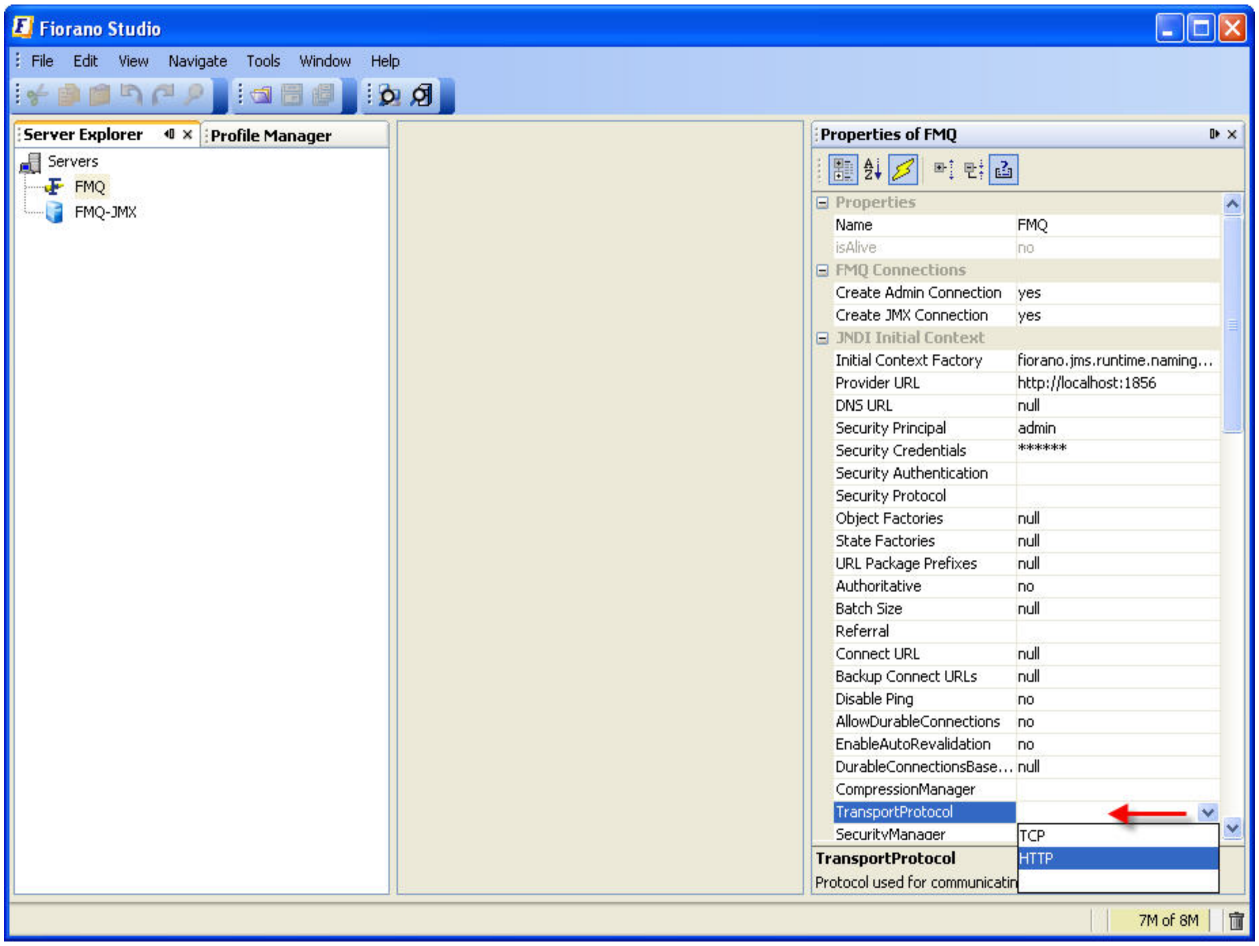
FMQ
No description available.

On successful login, an explorer window is displayed. This window displays all manageable components that are part of the FioranoMQ server.

25.1.2 Over HTTP Protocol

1. Specify connection properties described in section 25.1.1 Over Default Configuration .

2. Change the **TransportProtocol** property to **HTTP** that is provided as part of the **JNDI Initial Context** (as shown in the figure below).



Properties	
Name	FMQ
isAlive	no
FMQ Connections	
Create Admin Connection	yes
Create JMX Connection	yes
JNDI Initial Context	
Initial Context Factory	fiorano.jms.runtime.naming...
Provider URL	http://localhost:1856
DNS URL	null
Security Principal	admin
Security Credentials	******
Security Authentication	
Security Protocol	
Object Factories	null
State Factories	null
URL Package Prefixes	null
Authoritative	no
Batch Size	null
Referral	
Connect URL	null
Backup Connect URLs	null
Disable Ping	no
AllowDurableConnections	no
EnableAutoRevalidation	no
DurableConnectionsBase...	null
CompressionManager	
TransportProtocol	TCP / [✓ HTTP]
SecurityManager	

TransportProtocol
Protocol used for communicatin

3. Connect to the server by selecting the **Login** option displayed by right-clicking the menu.

On successful login, an explorer windowis displayed. This window displays all manageable components that are part of the FioranoMQ server in a tree format.

25.2 Working with Connection Factories

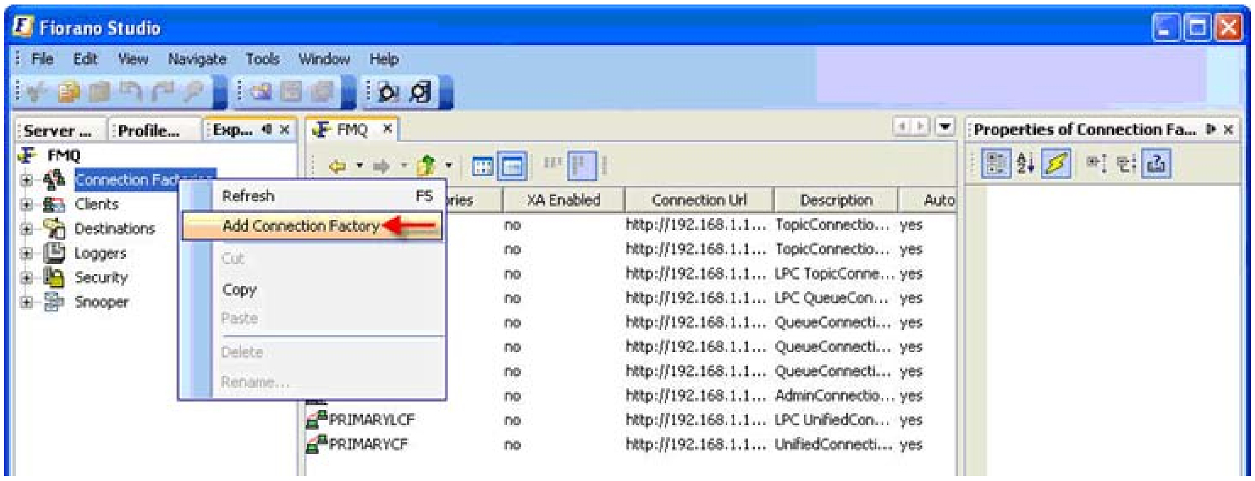
25.2.1 Adding a Connection Factory

To add a new connection factory, follow the steps given below:

1. Connect to the server as described in section 25.1.1 Over Default Configuration .

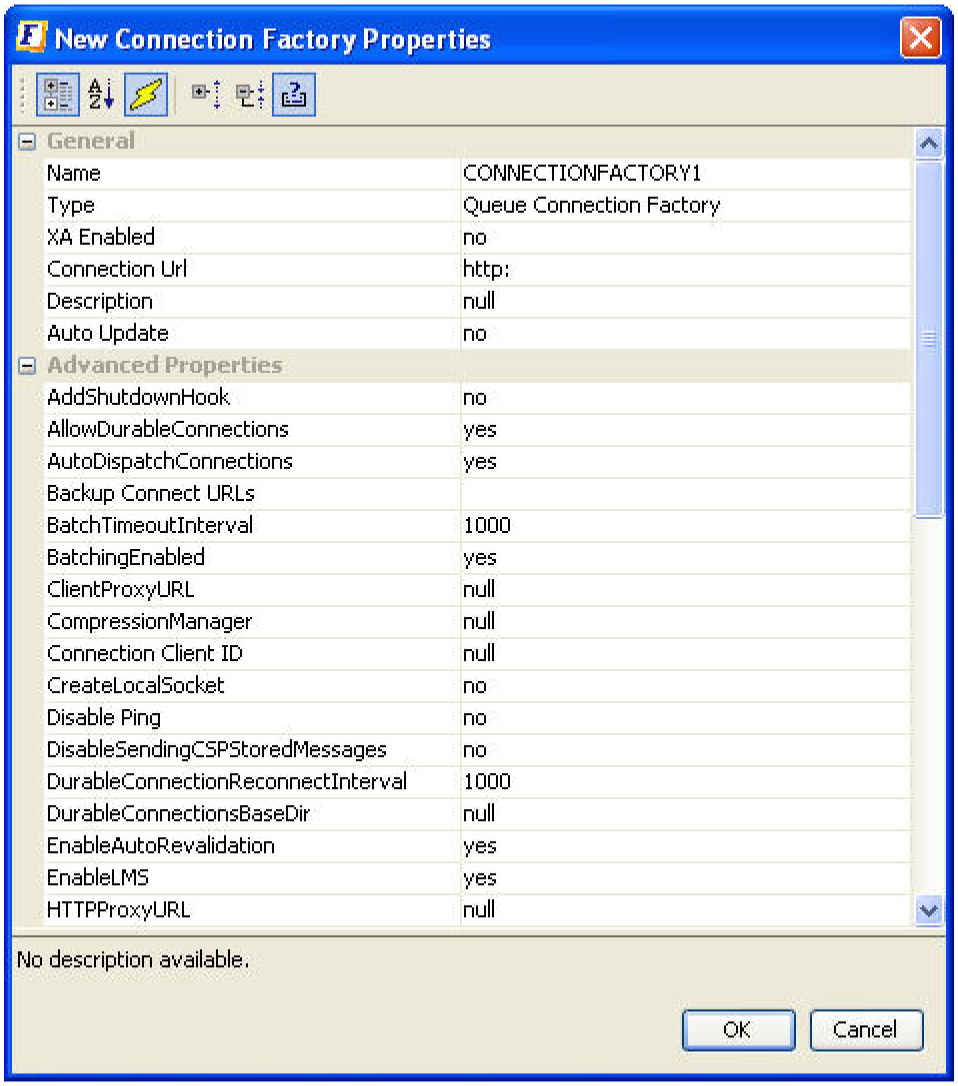
2. Select the **Connection Factories** node from the **Server Explorer**.

3. Right-click and select **Add Connection Factory** from the pop-up menu.



...ries	XA Enabled	Connection Url	Description	Auto
	no	http://192.168.1.1...	TopicConnectio...	yes
	no	http://192.168.1.1...	TopicConnectio...	yes
	no	http://192.168.1.1...	LPC TopicConne...	yes
	no	http://192.168.1.1...	LPC QueueCon...	yes
	no	http://192.168.1.1...	QueueConnecti...	yes
	no	http://192.168.1.1...	QueueConnecti...	yes
	no	http://192.168.1.1...	QueueConnecti...	yes
	no	http://192.168.1.1...	AdminConnectio...	yes
PRIMARYLCF	no	http://192.168.1.1...	LPC UnifiedCon...	yes
PRIMARYCF	no	http://192.168.1.1...	UnifiedConnecti...	yes

4. The **New Connection Factory Properties** dialog box is displayed. Configure these properties in accordance with the requirements of the application.



General	
Name	CONNECTIONFACTORY1
Type	Queue Connection Factory
XA Enabled	no
Connection Url	http:
Description	null
Auto Update	no
Advanced Properties	
AddShutdownHook	no
AllowDurableConnections	yes
AutoDispatchConnections	yes
Backup Connect URLs	
BatchTimeoutInterval	1000
BatchingEnabled	yes
ClientProxyURL	null
CompressionManager	null
Connection Client ID	null
CreateLocalSocket	no
Disable Ping	no
DisableSendingCSPStoredMessages	no
DurableConnectionReconnectInterval	1000
DurableConnectionsBaseDir	null
EnableAutoRevalidation	yes
EnableLMS	yes
HTTPProxyURL	null

No description available.

5. Click the **OK** button and a new Connection Factory with the specified name is created.

25.2.2 Deleting a Connection Factory

Unwanted connection factories can be deleted by executing the steps below:

1. Select the **Connection Factories** node from the **Server Explorer**.

2. Select the Connection Factory name and right-click the mouse to select **Delete** from the pop-up menu. A **Confirm Object Deletion** dialog box is displayed. Click the **Yes** button in this dialog box.

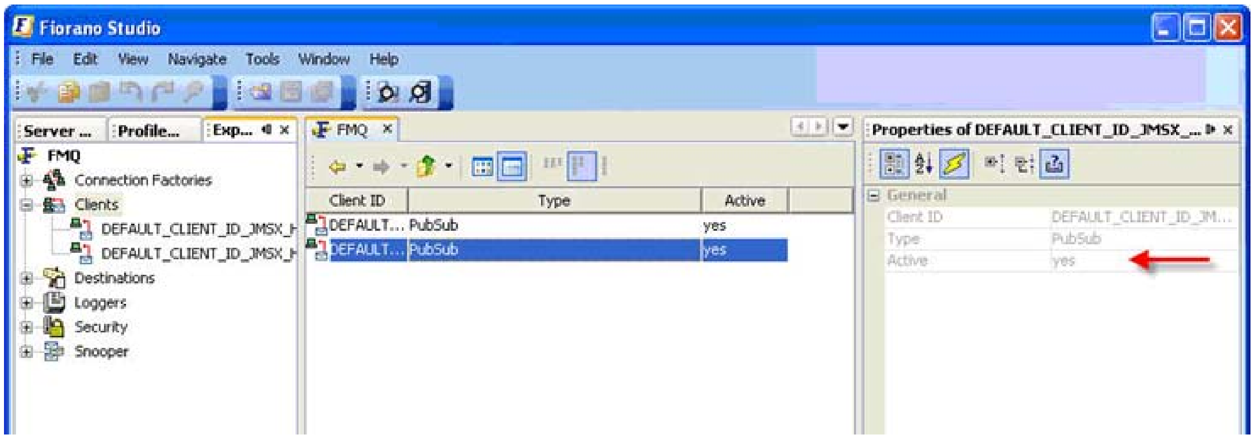
25.3 Working with Durable Subscriptions

The **Durable Subscriptions** node displays the list of client IDs (of the Durable Subscribers) that exist on the server. The FioranoMQ Administrator can unsubscribe or delete all pending messages of an inactive durable subscriber through this node.

25.3.1 Viewing Durable Subscriptions

1. The **Client** node in the profile tree corresponds to the Durable Subscribers. It lists the offline and the online Client IDs on which the Durable Subscription currently running.

2. To view the Durable Subscribers, run the sample PubSub/ DurableSubscribers and refresh the Client node. This displays the list of clients as shown in the figure below. On selecting **Clients**, the properties pane displayswhether the client is Active or Inactive.

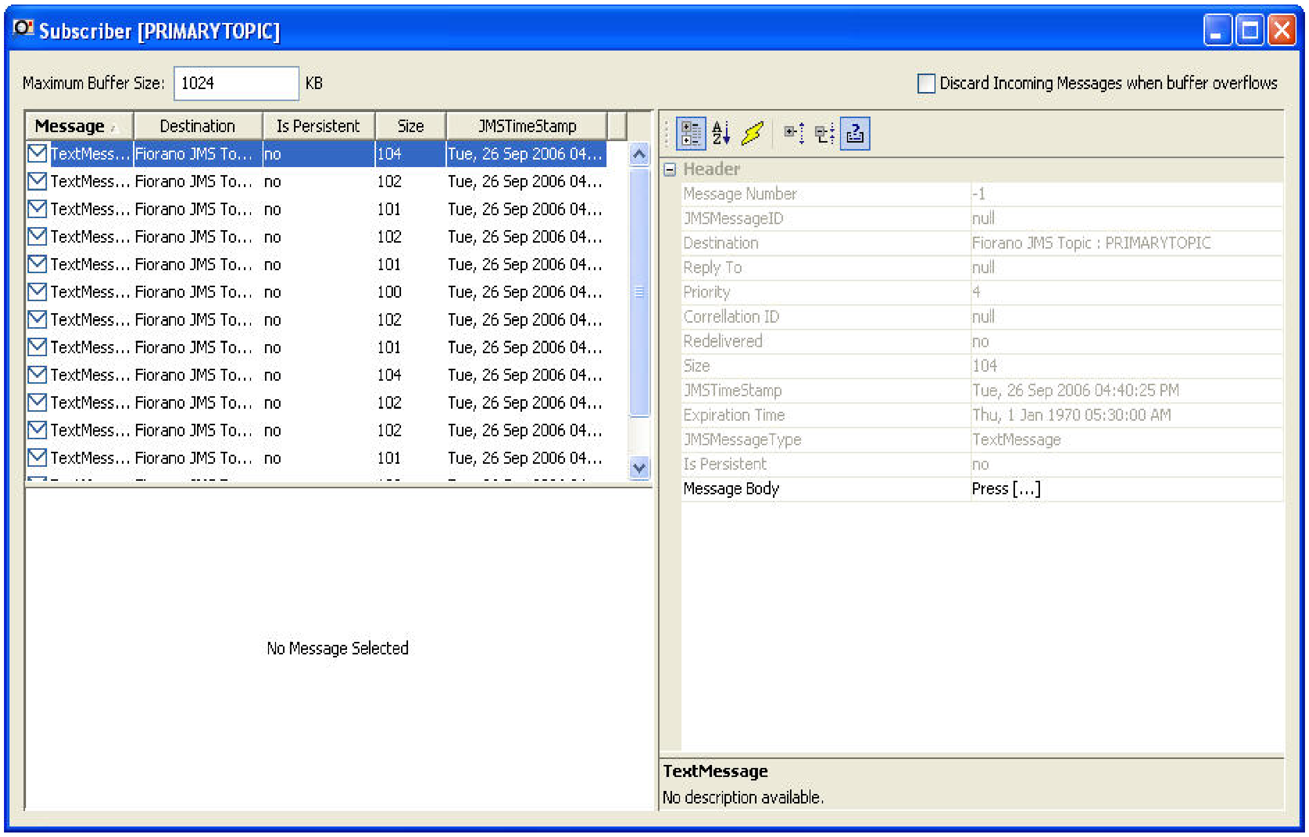


Client ID	Type	Active
DEFAULT...	PubSub	yes
DEFAULT...	PubSub	yes

General	
Client ID	DEFAULT_CLIENT_ID_JM...
Type	PubSub
Active	yes

3. Double-click the Client ID to view the Subscription ID. (Shut down the DurableSubscriber sample and not the Publisher). You can send messages from the Publisher. Now right-click the SubscriptionID and select Browse messages.

4. The following window appears with the messages sent from the Publisher.


Subscriber [PRIMARYTOPIC]

Maximum Buffer Size: 1024 KB

Discard Incoming Messages when buffer overflows

Message	Destination	Is Persistent	Size	JMSTimeStamp
TextMess...	Fiorano JMS To...	no	104	Tue, 26 Sep 2006 04...
TextMess...	Fiorano JMS To...	no	102	Tue, 26 Sep 2006 04...
TextMess...	Fiorano JMS To...	no	101	Tue, 26 Sep 2006 04...
TextMess...	Fiorano JMS To...	no	102	Tue, 26 Sep 2006 04...
TextMess...	Fiorano JMS To...	no	101	Tue, 26 Sep 2006 04...
TextMess...	Fiorano JMS To...	no	100	Tue, 26 Sep 2006 04...
TextMess...	Fiorano JMS To...	no	102	Tue, 26 Sep 2006 04...
TextMess...	Fiorano JMS To...	no	101	Tue, 26 Sep 2006 04...
TextMess...	Fiorano JMS To...	no	104	Tue, 26 Sep 2006 04...
TextMess...	Fiorano JMS To...	no	102	Tue, 26 Sep 2006 04...
TextMess...	Fiorano JMS To...	no	102	Tue, 26 Sep 2006 04...
TextMess...	Fiorano JMS To...	no	101	Tue, 26 Sep 2006 04...

No Message Selected

Header

Message Number	-1
JMSMessageID	null
Destination	Fiorano JMS Topic : PRIMARYTOPIC
Reply To	null
Priority	4
Correllation ID	null
Redelivered	no
Size	104
JMSTimeStamp	Tue, 26 Sep 2006 04:40:25 PM
Expiration Time	Thu, 1 Jan 1970 05:30:00 AM
JMSMessageType	TextMessage
Is Persistent	no
Message Body	Press [...]

TextMessage

No description available.

25.3.2 Purging Messages of the Durable Subscriptions

The administrator can delete pending messages for all subscribers of the selected clientID. To delete pending messages, execute the following steps:

1. Select the **Durable Subscriptions** node from the **Server Explorer** pane.
2. Select the ClientID shown under the **Durable Subscriptions** node.
3. Right-click and select the **Purge** option from the pop-up menu.

25.3.3 Refreshing Durable Subscriptions

The list of **Durable Subscriptions** can be refreshed by following the following steps:

1. From the **Server Explorer**, select the **Durable Subscriptions** node.
2. Right-click and select the **Refresh** option from the pop-up menu.

25.4 Working with Destinations

The Destination node lists all the destinations created on the server. The destination node allows the administrator to create or delete a destination from the server.

25.4.1 Managing Topics

The Topics node displays the list of topics on the server. The administrator can create or remove a topic, as well as edit the ACL of a topic. By default, all Users on the server possess the required permission. A user can assign Publish, Subscribe, Durable Subscribe and Unsubscribe permission statuses to Users or User Groups.

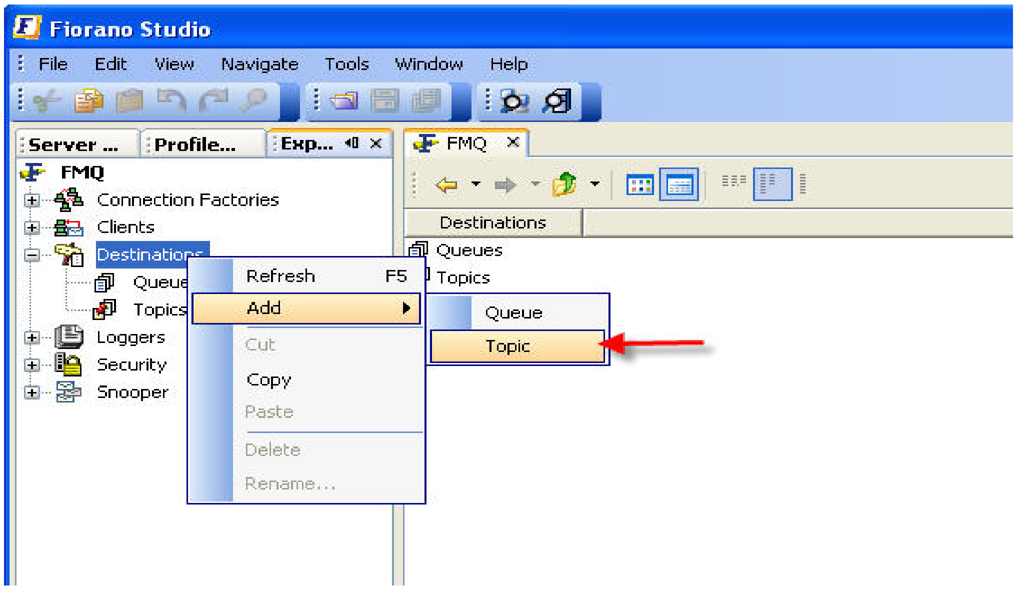
25.4.2 Adding a New Topic

The Server Explorer pane shows the Destinations node, which, on expanding, shows the Topics sub-node. The Topics sub-node displays the list of all the topics. The administrator can create topics and manage them through the Server Explorer. A User can create new topics using two methods, both of which are described below:

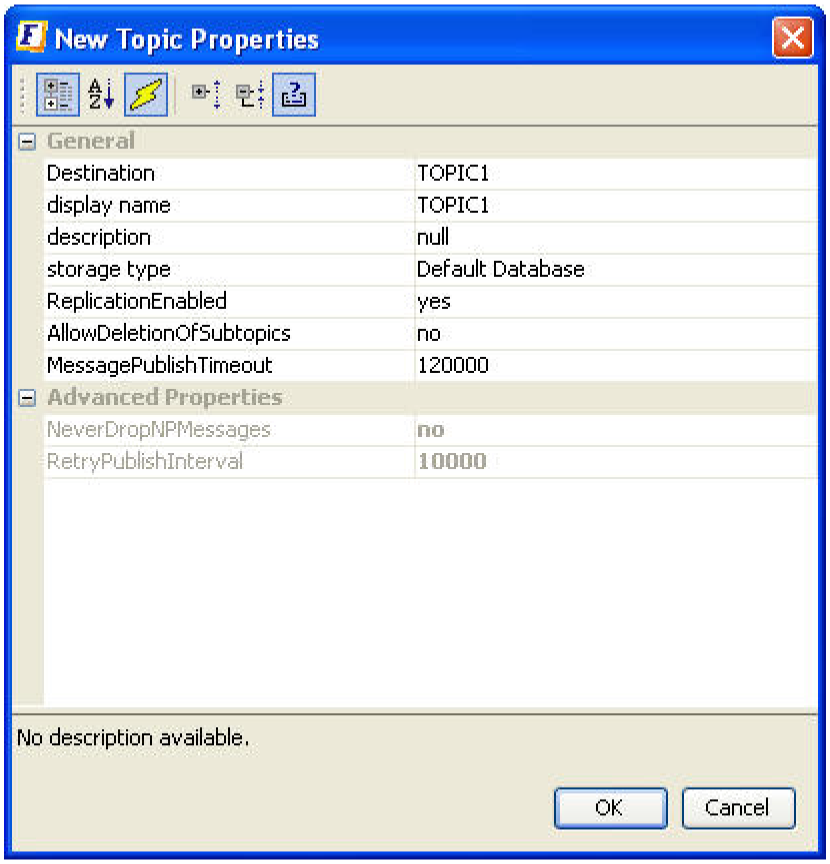
Method 1: From the Destinations node

The Server Explorer pane shows the Destinations node, which enables the administrator to add new topics to the existing topic list. Follow the steps given below to create a new topic:

1. From the Server Explorer, select the Destinations node.
2. Right-click and select **Add > topic** from the pop-up menu.



3. The **New Properties** dialog box is displayed. Configure the properties as required.



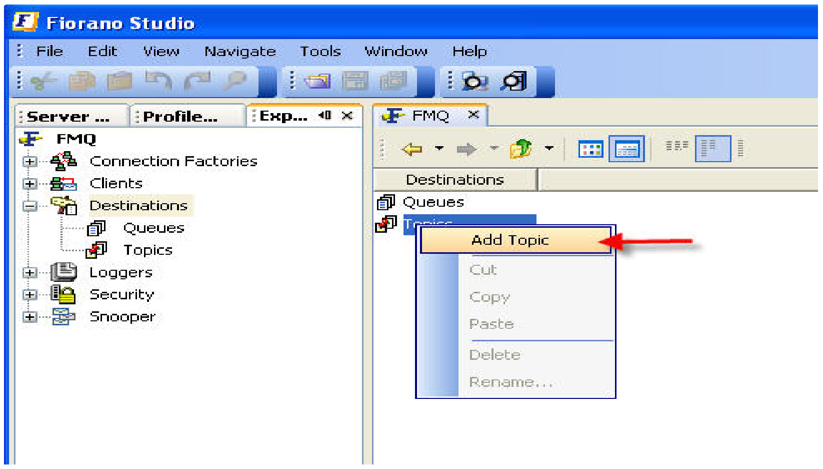
General	
Destination	TOPIC1
display name	TOPIC1
description	null
storage type	Default Database
ReplicationEnabled	yes
AllowDeletionOfSubtopics	no
MessagePublishTimeout	120000
Advanced Properties	
NeverDropNPMessages	no
RetryPublishInterval	10000

No description available.

4. Click the **OK** button. The new topic TOPIC1 will be added to the list of topics with the specified properties.

Method 2: From the Topics sub-node

1. From the **Topics** sub-node, right-click and select **Add topic** from the pop-up menu.



2. The **New Topic Properties** dialog box is displayed (as previously shown). Enter the relevant information.

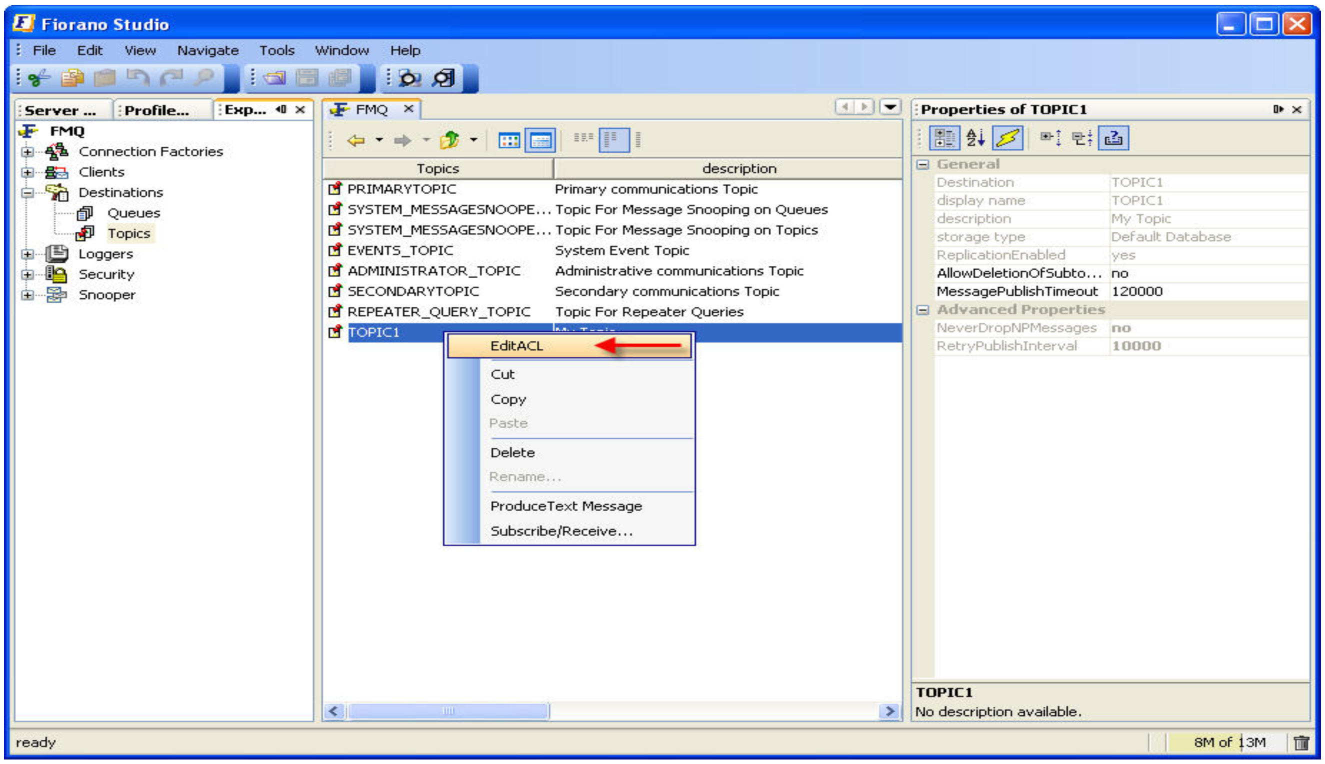
3. Click the **OK** button. The new topic will be added to the list of topics with specified properties (as previously shown).

25.4.3 Editing Access Control List (ACL)

Once a topic is created, the Access Control List (ACL) for the topic must be created. To created the ACL, follow the steps below:

1. Select the **Topics** sub-node. Select the topic name from the list of topics.

2. Right-click and select **Edit ACL** option from the pop-up menu.

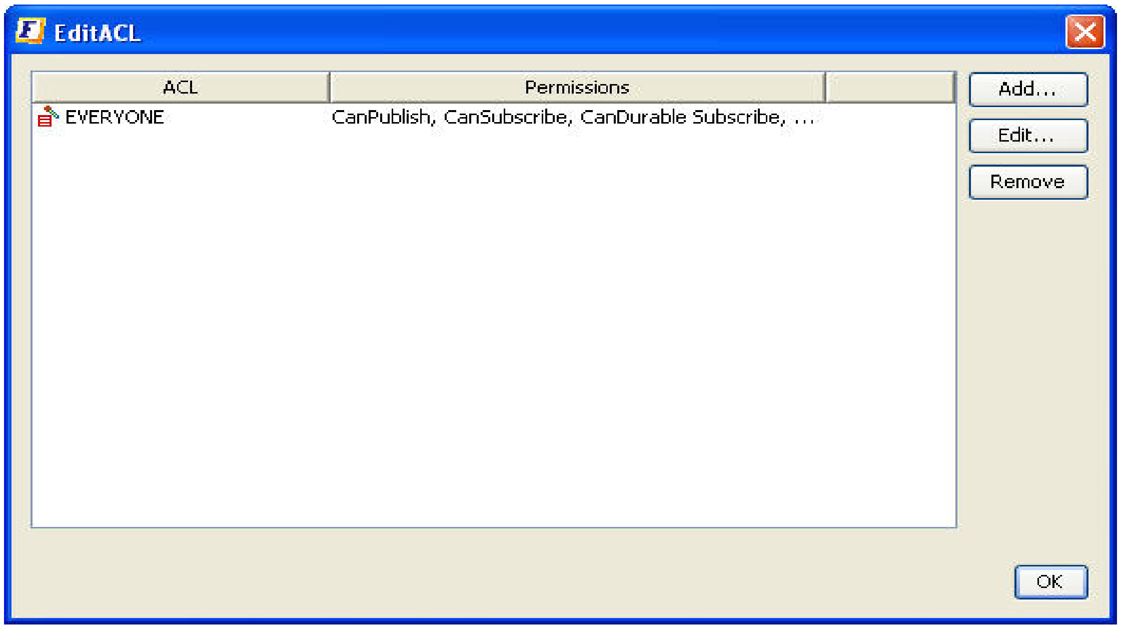


Topics	description
PRIMARYTOPIC	Primary communications Topic
SYSTEM_MESSAGESNOOPE...	Topic For Message Snooping on Queues
SYSTEM_MESSAGESNOOPE...	Topic For Message Snooping on Topics
EVENTS_TOPIC	System Event Topic
ADMINISTRATOR_TOPIC	Administrative communications Topic
SECONDARYTOPIC	Secondary communications Topic
REPEATER_QUERY_TOPIC	Topic For Repeater Queries
TOPIC1	My Topic

General	
Destination	TOPIC1
display name	TOPIC1
description	My Topic
storage type	Default Database
ReplicationEnabled	yes
AllowDeletionOfSubto...	no
MessagePublishTimeout	120000
Advanced Properties	
NeverDropNPMessages	no
RetryPublishInterval	10000

TOPIC1
No description available.

3. The Edit ACL dialog box appears and shows various entries in the ACL of the topic. Each entry corresponds to a principal, which can consist of either a User or a Group. Each entry is associated with a set of permissions, which can be either **allow**, **disallow** or **default**. An allow permission implies that the principal is allowed the set of selected permissions. A disallow permission implies that the principal is denied the set of permissions mentioned in the entry. If no permission is manually assigned to the principal, then the permission is set to default. By default, the ACL for a topic has the group 'everyone' in its list with all the permissions set to 'allow'. Since the Group 'everyone' includes all the principals created in the FioranoMQ Server, the topic assigns all permissions to all principals. The following set of permissions can be assigned to a topic:
   - Publish
   - Subscribe
   - Unsubscribe
   - Durable Subscribe



ACL	Permissions	
EVERYONE	CanPublish, CanSubscribe, CanDurable Subscribe, ...	

4. To change the permissions assigned to a member subscribing to a particular topic, select the member and Click the **Edit** button. The **Edit Permissions** dialog box appears. Assign the required permissions and click the **OK** button.
5. To add a member to a particular topic, click the **Add** button on the ACL dialog box. The **Add New ACL Entry** dialog box appears. Select the member name and click the **OK** button.
6. To remove a member from a particular topic, select the member name from the **Edit ACL** dialog box and click the **Remove** button.
7. Click the **OK** button.

25.4.4 Removing a Topic

The administrator can remove the topics that are not required from the topic list. To remove a topic from the topics list, follow the steps below:

1. From the **Server Explorer**, select the **Destinations** node.
2. Select the **Topics** sub-node. Select the Topic name to be deleted and right-click the mouse. Select **Delete** from the pop-up menu.
3. The **Confirm Object Deletion** dialog box is displayed. Click the **Yes** button.
4. The selected topic has been deleted and is therefore not visible in the topic list.

Multiple topics can be removed by selecting the required topic (by pressing the **Ctrl** key), and then pressing the **Del** key.

25.4.5 Managing Queues

The Queues node displays the list of queues on the server. The Administrator can create and remove the Queues and edit the ACL. By default, all Users on the server have all permissions. The Administrator can assign Send, Receive and Browse permissions to Users and Groups. Browsing the Queue allows the administrator to view the total number of deleted messages, undeleted messages and message properties. Undeleted messages are those messages for which the server has not received acknowledgment from a client.

25.4.6 Adding a New Queue

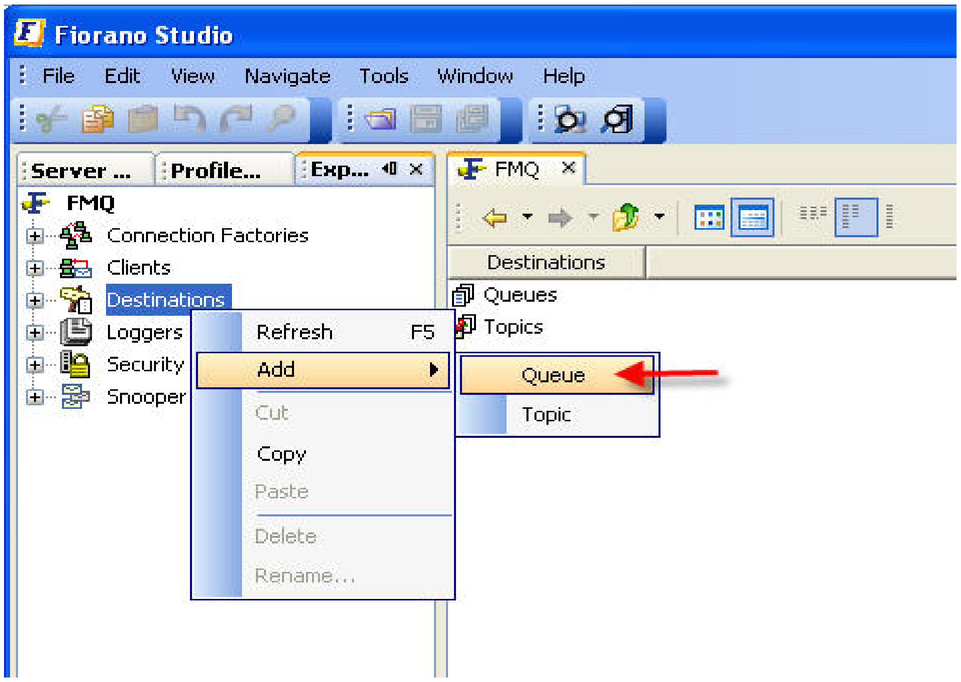
The **Server Explorer** pane shows the **Destinations** node, which, on expanding, shows the **Queues** sub-node. The **Queues** sub-node displays the list of all the queues. The administrator can create queues and manage them through the **Server Explorer**.

The administrator can create new queues by using the two methods described below:

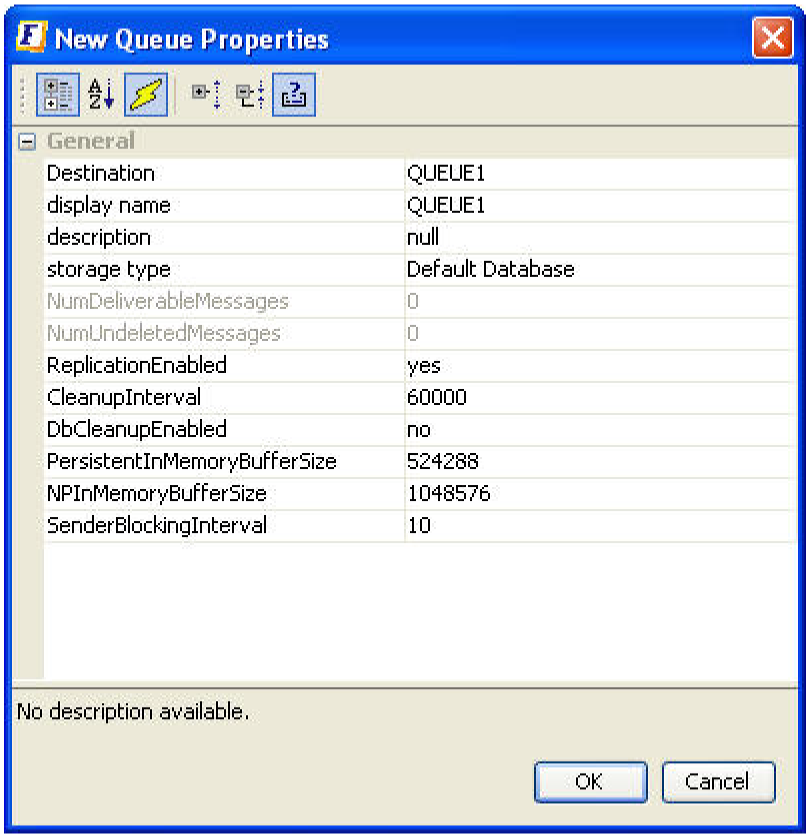
Method 1: From the Destinations node

The **Server Explorer** pane shows the **Destinations** node, which enables the administrator to add new queues to the existing queue list. Follow the steps below to create a new queue:

1. From the **Server Explorer**, select the **Destinations** node.
2. Right-click and select **Add > queue** from the pop-up menu.



3. The **New Queue Properties** dialog box appears. Configure properties as required.


New Queue Properties

General

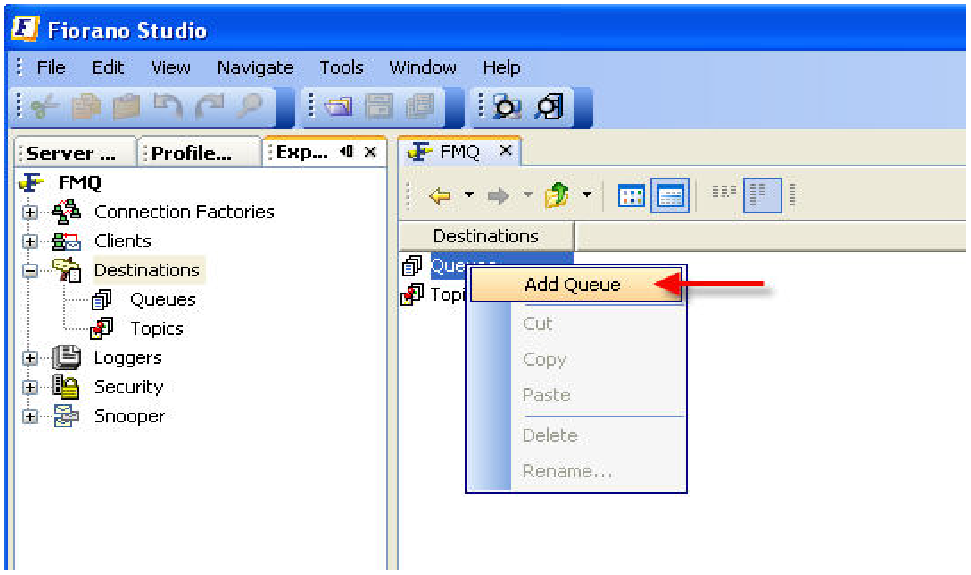
Destination	QUEUE1
display name	QUEUE1
description	null
storage type	Default Database
NumDeliverableMessages	0
NumUndeletedMessages	0
ReplicationEnabled	yes
CleanupInterval	60000
DbCleanupEnabled	no
PersistentInMemoryBufferSize	524288
NPInMemoryBufferSize	1048576
SenderBlockingInterval	10

No description available.

4. Click the **OK** button. The new queue will be added to the list of queues with specified properties.

Method 2: From the Queues sub-node

1. From the **Queues** sub-node, right-click and select **Add Queue** from the pop-up menu.

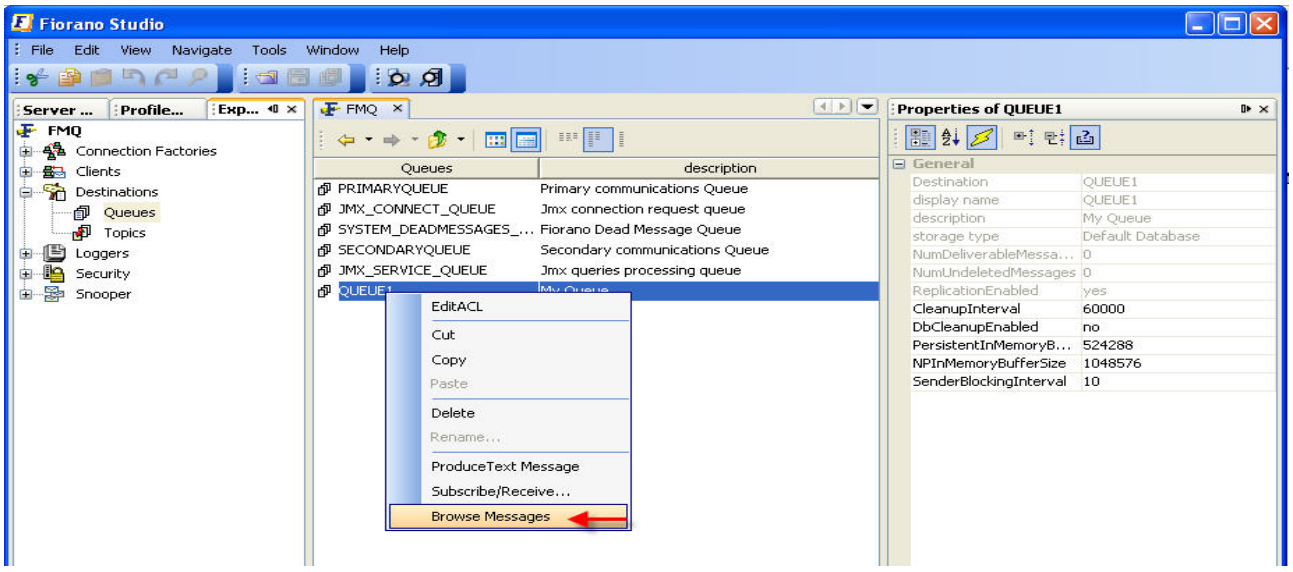


2. The **New Queue Properties** dialog box is displayed. Enter the relevant information.
3. Click the **OK** button. The new queue will be added to the list of queues with specified properties.

25.4.7 Browsing a Queue

Browsing a queue allows the administrator to view all the messages that are present in the queue. Follow the steps below to browse a queue:

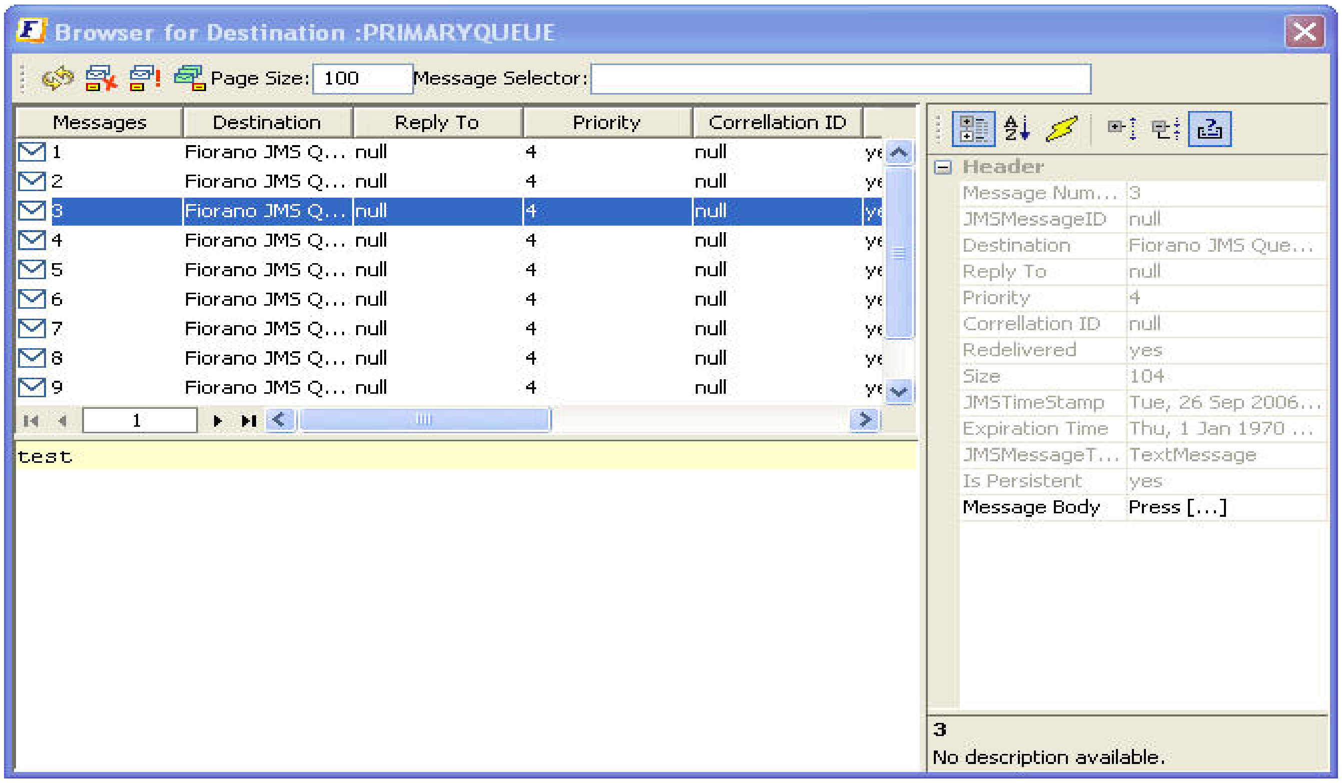
1. Select the **Queues** sub-node. Select the name of the queue to be browsed from the **Queues** list.
2. Right-click and select **Browse Messages** from the pop-up menu.



Queues	description
PRIMARYQUEUE	Primary communications Queue
JMX_CONNECT_QUEUE	Jmx connection request queue
SYSTEM_DEADMESSAGES_...	Fiorano Dead Message Queue
SECONDARYQUEUE	Secondary communications Queue
JMX_SERVICE_QUEUE	Jmx queries processing queue
QUEUE1	My Queue

General	
Destination	QUEUE1
display name	QUEUE1
description	My Queue
storage type	Default Database
NumDeliverableMessa...	0
NumUndeletedMessages	0
ReplicationEnabled	yes
CleanupInterval	60000
DbCleanupEnabled	no
PersistentInMemoryB...	524288
NPInMemoryBufferSize	1048576
SenderBlockingInterval	10

3. The **Browser for Destination** dialog box is displayed. Send the messages through PTP samples and open the browser window.



Browser for Destination :PRIMARYQUEUE

Page Size: 100 Message Selector:

Messages	Destination	Reply To	Priority	Correllation ID	
1	Fiorano JMS Q...	null	4	null	ye
2	Fiorano JMS Q...	null	4	null	ye
3	Fiorano JMS Q...	null	4	null	ye
4	Fiorano JMS Q...	null	4	null	ye
5	Fiorano JMS Q...	null	4	null	ye
6	Fiorano JMS Q...	null	4	null	ye
7	Fiorano JMS Q...	null	4	null	ye
8	Fiorano JMS Q...	null	4	null	ye
9	Fiorano JMS Q...	null	4	null	ye

1

test

Header

Message Num...	3
JMSMessageID	null
Destination	Fiorano JMS Que...
Reply To	null
Priority	4
Correllation ID	null
Redelivered	yes
Size	104
JMSTimeStamp	Tue, 26 Sep 2006...
Expiration Time	Thu, 1 Jan 1970 ...
JMSMessageT...	TextMessage
Is Persistent	yes
Message Body	Press [...]

3
No description available.

4. The Browser also contains a **Message Selector**. It allows a client to specify, by header field references and property references, the messages it wishes to view/access. Only messages whose header and property values match those of the selector can be browsed. A message selector matches a message if the selector arrives at 'true' when the message header field values and property values are substituted by their corresponding identifiers in the selector. If the value of a message selector is an empty string, the value is treated as null and indicates that there is no message selector for the message consumer.

5. The message properties can be viewed from the **Properties** pane of the Browser.

6. Double-click the message to open the message body.

7. Close the Browser by clicking on the **Close** (**X**) button on the top right of the window.

25.4.8 Editing Access Control List (ACL) and Removing a Queue

The procedure is the same as previously explained with reference to Topics, except that, here, in place of Topics it will now be applied to Queues.

25.4.9 Setting and Configuring OnTheFlyCreationOfDestinations

OnTheFlyCreationOfDestinations property controls the behavior of createTopic and createQueue APIs' in a JMS Session. By enabling this property, when the createTopic/createQueue APIs are called on non-existent Destinations, Destinations will be created for these APIs.

If the OnTheFlyCreationOfDestinations property is disabled, thenADMINISTERED_OBJECT_DOESNOT_EXISTS exception is thrown if a destination does not exist. If a destination is present, then it acts as a lookup method. This feature is controlled by the flag AllowOnTheFlyCreationOfDestinations

The AllowOnTheFlyCreationOfDestinations can be enabled/disabled by following the steps below:

1. Open the profile for off-line editing through the Profile Manager as explained in section 4.8.1
2. Go to **Fiorano -> etc -> FMQConfigLoade**r.
3. Turn on **AllowOnTheFlyCreationOfDestinations** property by changing its value to **yes**.

Note: This flag does not affect the creation of destinations from **Admin Services**. Even though this flag is disabled, Destinations can be created by the Admin User using Admin connection and Admin Services.

25.5 Working with Security

The administrator can create, manage and manipulate Users and Groups through the Security node. The administrator can change User passwords as well as add Users or Groups to other existing Groups.

25.5.1 Managing Users

The Users sub-node displays the list of all Users whose identity is known to the server. Fiorano has admin, ayrton, and anonymous as its default users. The Administrator can create and delete Users, as well as change their passwords. User properties can be viewed through the Properties pane.

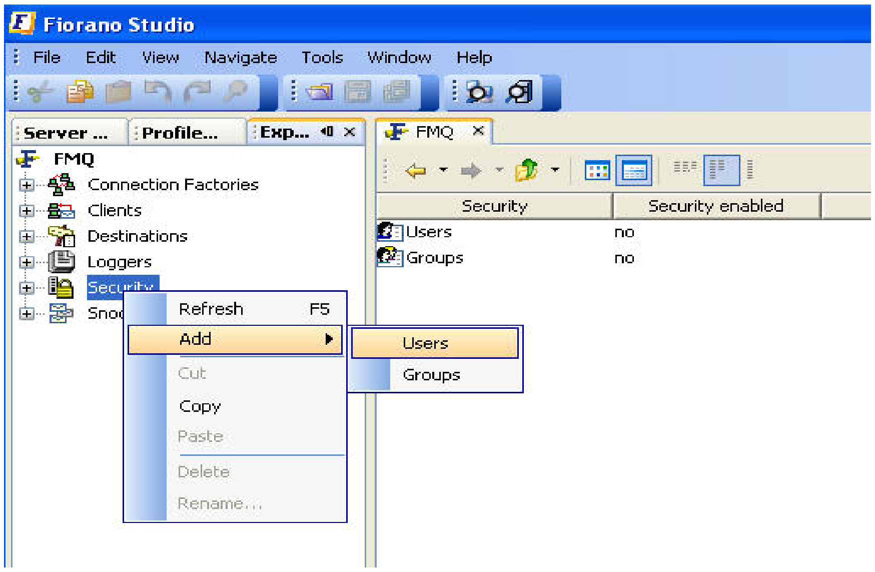
25.5.2 Adding a New User

The administrator can create Users and manage them through the **Server Explorer**. The administrator can create new Users through the **Security** node as well as through the **Users** sub-node. Clicking on the **Users** sub-node displays the list of all Users whose identity is known to the server along with their connection status. The connection status informs the administrator whether or not that particular User is currently connected to the FioranoMQ Server. The administrator can create new Users by employing either of the methods described below:

Method 1: From the Security node

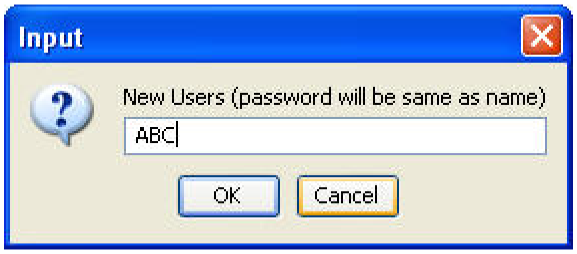
The **Server Explorer** pane shows the **Security** node, which enables an administrator to add new Users to the existing User list. To create a new User, follow the steps below:

1. Select the **Security** node from the **Server Explorer**.
2. Right-click and select **Add > user** from the pop-up menu.



Security	Security enabled
Users	no
Groups	no

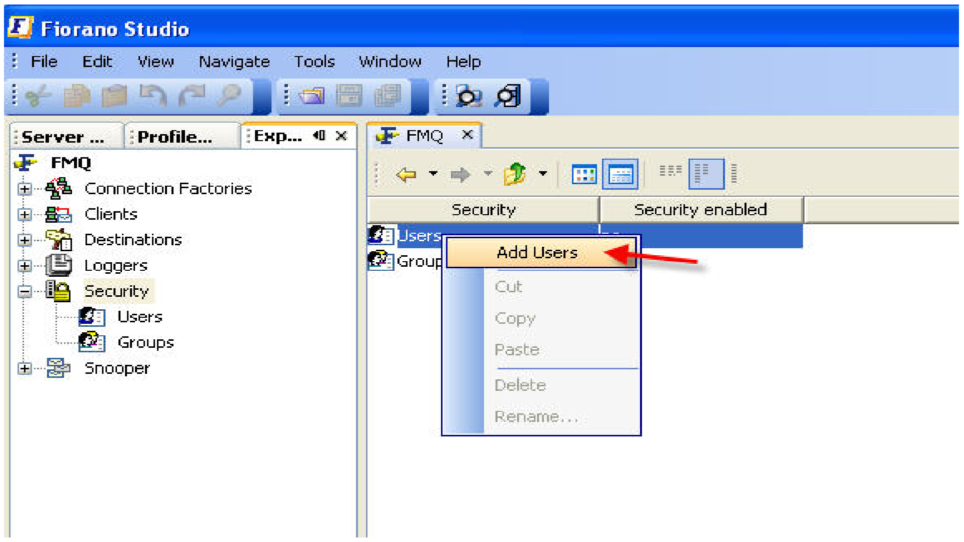
3. The Admin Studio **Input** dialog box is displayed. Enter the **New User Name** and click the **OK** button.



A new User is created with the password as the username of the User.

Method 2: From the Users sub-node

1. From the **Users** sub-node, right-click and select **Add user** from the pop-up menu.

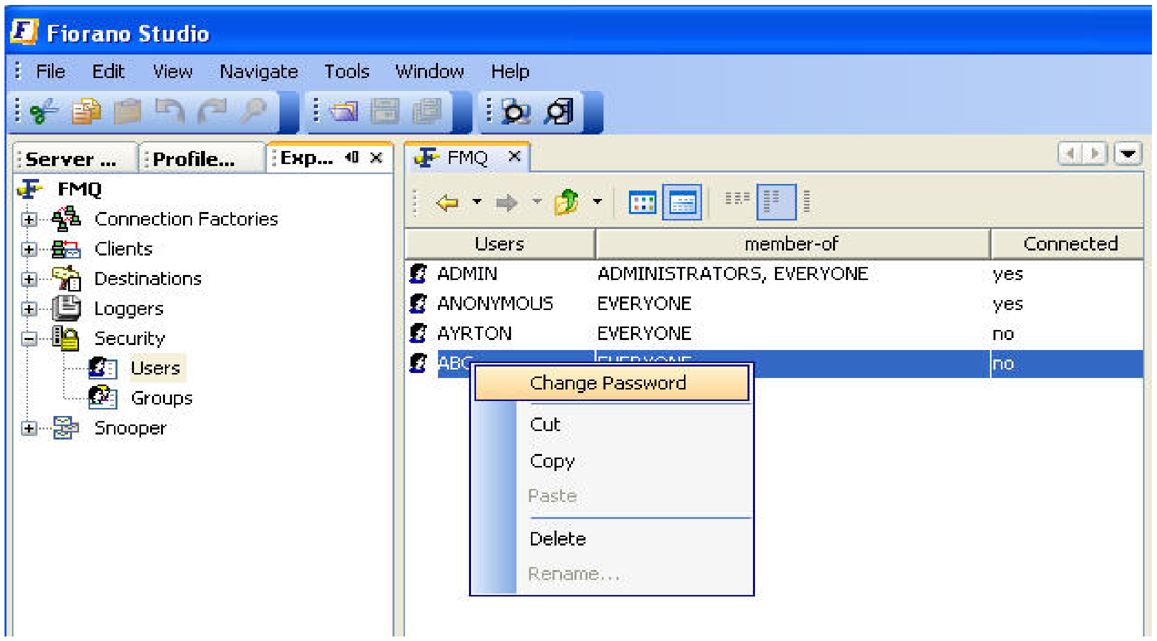


2. The Admin Studio Input dialog box is displayed. Enter the Username and click the **OK** button. The new User will be added to the list of pre-existing Users.

25.5.3 Changing User Password

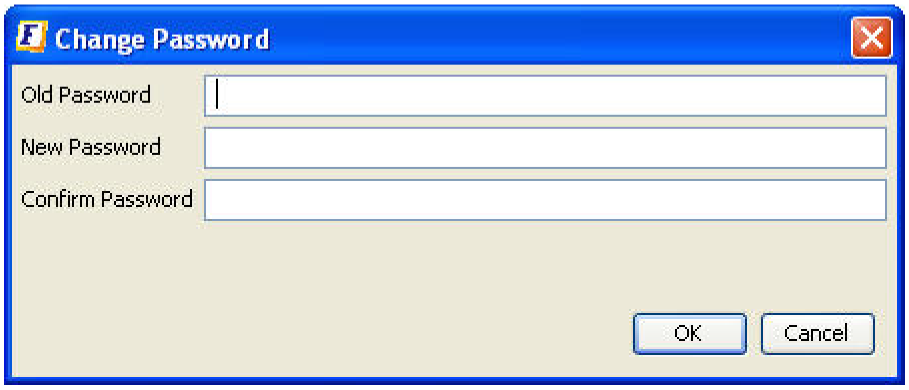
The administrator can change the User password by following the steps given below:

1. Select the **Users** sub-node. Select the name of the User whose password is to be changed and right-click the mouse.
2. Select **Change Password** from the pop-up menu and the **Change Password** dialog is displayed.



Users	member-of	Connected
ADMIN	ADMINISTRATORS, EVERYONE	yes
ANONYMOUS	EVERYONE	yes
AYRTON	EVERYONE	no
ABC	EVERYONE	no

3. Provide the old password and the new password and click the **OK** button.



25.5.4 Removing a User

The administrator can remove Users from the User list.

To remove a User, follow the steps below:

1. Select the **Users** sub-node from the **Server Explorer** pane.
2. Select the name of the user to be deleted. Right-click and select **Delete** from the pop-up menu.
3. The **Confirm Object Deletion** dialog box is displayed. Click the **Yes** button.

The selected User is deleted and is removed from the list of Users.

25.5.5 Managing Groups

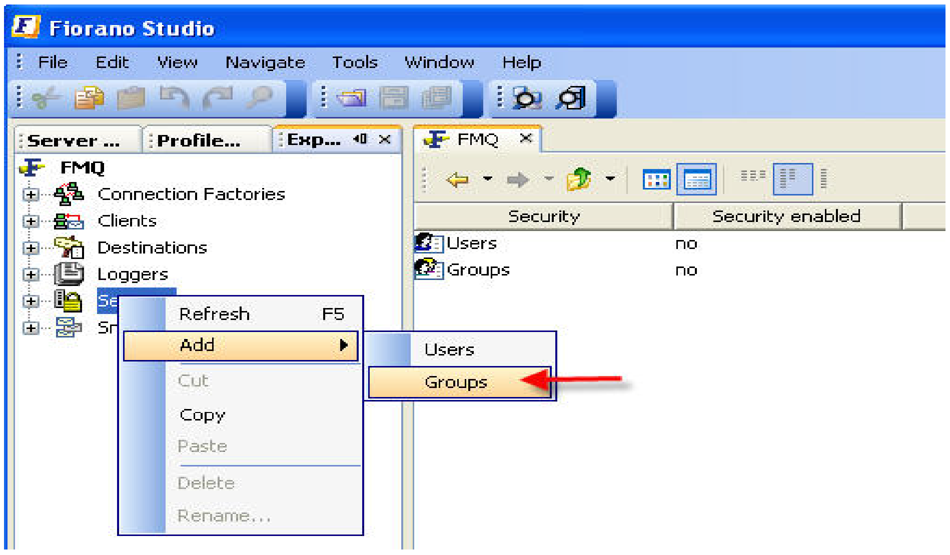
The Groups sub-node displays the list of Groups on the server. The administrator can create or remove a Group. By default, all Users and Groups are members of the '**everyone**' group.

25.5.6 Adding a New Group

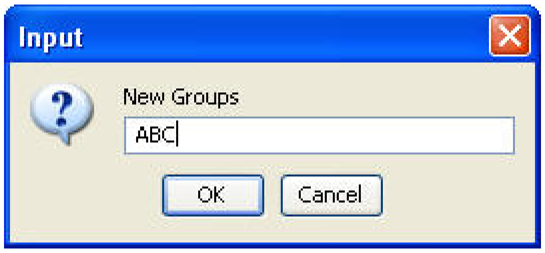
The administrator can create Groups and manage them through the **Server Explorer**. The **Server Explorer** pane shows the **Security** node, that enable the administrator to add new Groups to the existing Group list. FioranoMQ, by default, is bundled with a special group called **everyone**, which includes all users as its members. The **Groups** sub-node displays the list of Groups on the server. The administrator can create or remove a Group. The administrator can create a new group by employing either of the methods described below:

Method 1: From the Security node:

1. Select the **Security** node from the **Server Explorer**.
2. Right-click and select **Add > group** from the pop-up menu.



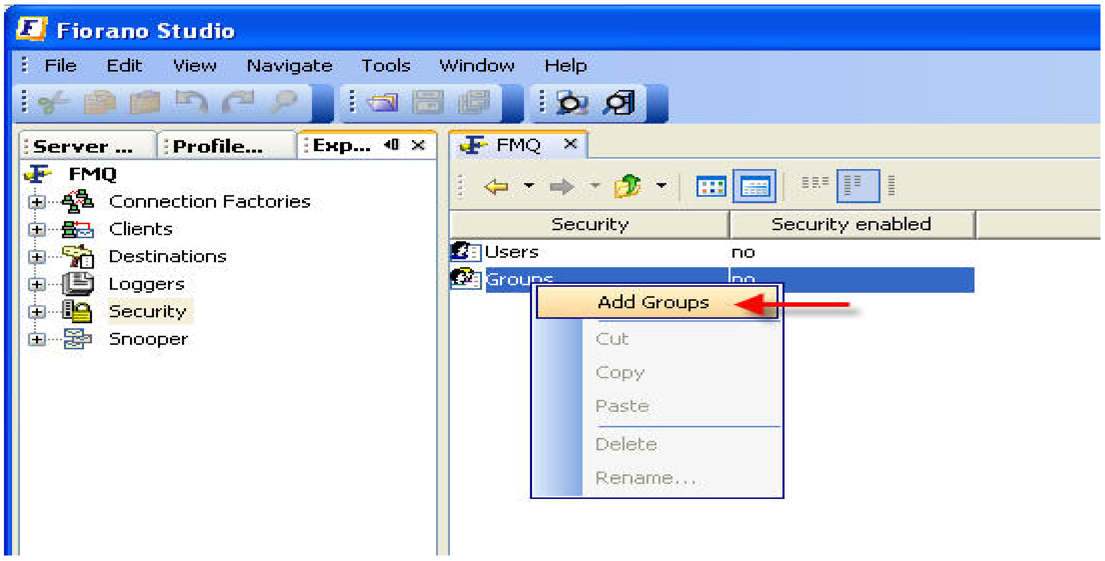
3. The Admin Studio **Input** dialog box is displayed.



4. Enter the Group name and click the **OK** button. The new Group is added to the Group list .

Method 2: From the Groups sub-node:

1. From the **Groups** sub-node, right-click and select **Add Group** from the pop-up menu.



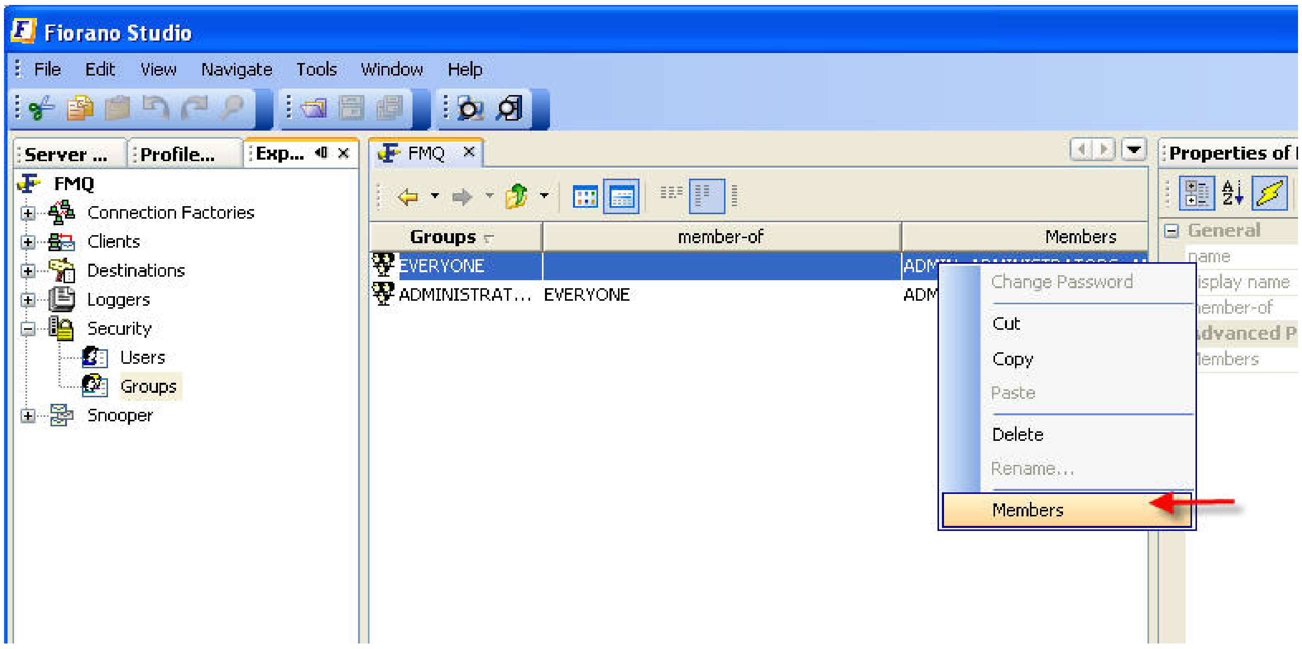
2. The Admin Studio **Input** dialog box is displayed.

3. Enter the Group name and click the **OK** button. The new Group is added to the list of Groups.

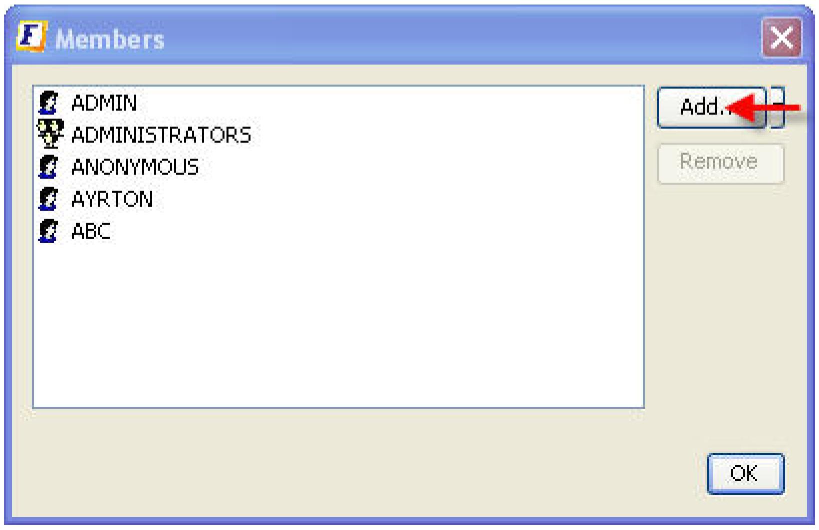
25.5.7 Adding a Member to a Group

The administrator can add a User or a Group as a member to an existing Group. To add a User or a Group to an existing Group, follow the steps below:

1. Select the **Groups** sub-node from the **Server Explorer** pane.
2. From the **Properties** pane, go to the **members** property. Click the ellipsis to show the members dialog box. The same dialog can be opened by double clicking on the **Groups** node.



3. Select the **Add** button to show the names of all Groups and Users that are not present in the Group.



4. Select the Users and/or Groups to be added to this Group and click the **OK** button.
5. Click the **Close** button.

25.5.8 Removing Member from Group

The administrator can remove a User or a Group member from an existing Group by following the steps below:

1. Select the **Groups** sub-node.
2. From the **Properties** pane, go to the member's property. Click the ellipsis to display the members dialog box. The same dialog can be opened by double clicking on the **Groups** node.
3. Select the name of the User to be removed from the Group. Click the **Remove** button.
4. Click the **Close** button.

25.5.9 Removing a Group

The administrator can remove Groups from the Group list.

To remove Groups from the Group list follow the steps below:

1. Select the **Groups** sub-node. Select the name of the group to be deleted and right-click the mouse.
2. Select **Delete** from the pop-up menu.
3. The **Confirm Object Deletion** dialog box is displayed. Click the **Yes** button.

The selected Group is deleted and is no longer visible in the Group list.

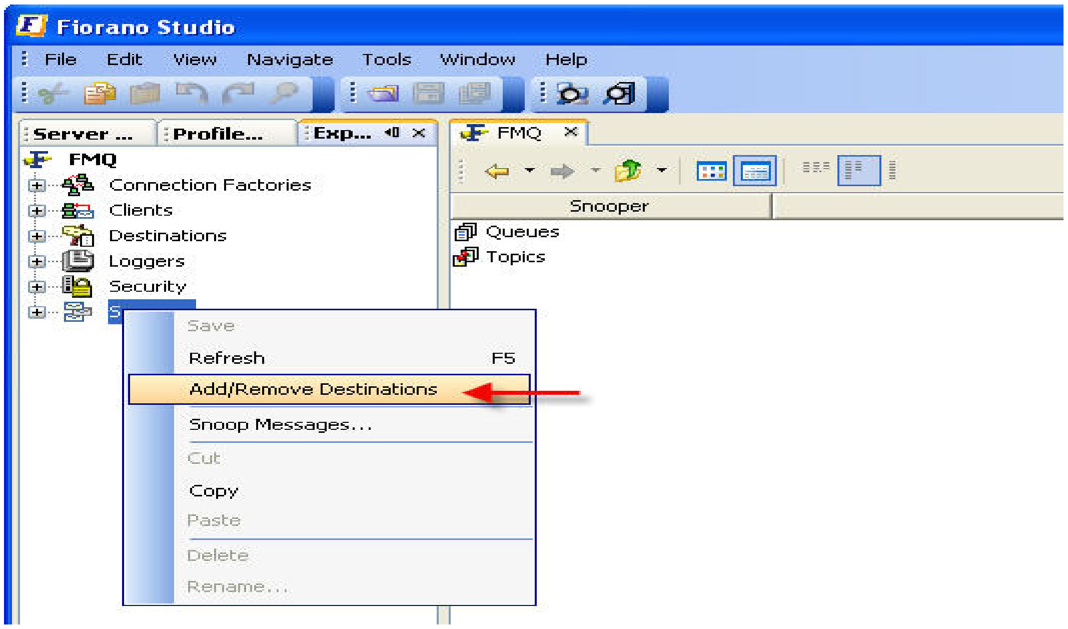
25.6 Working with Snooper

The administrator can exercise full control over the snooper system through the Fiorano Admin Studio. The snooper node allows the administrator to configure the FioranoMQ Server to snoop for messages on registered topics. The administrator can select topics to snoop on. The node displays messages that are snooped along with the destinations on which they were published. With the Snooper it is possible to snoop messages that are being sent to a destination. Through this node it is possible to view messages flowing through FioranoMQ. In addition, various parameters of the snooper subsystem can be configured as per requirements.

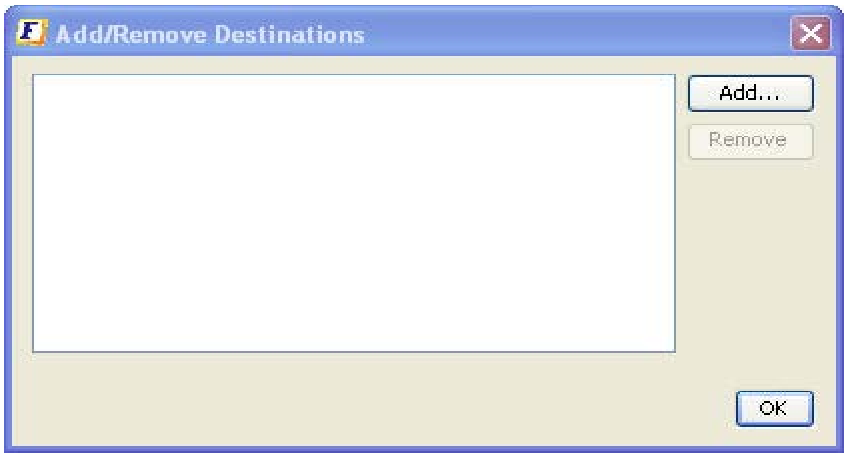
25.6.1 Adding Destinations in Snooper

With the Admin Studio it is possible to configure the FioranoMQ Server to snoop for messages on registered destinations. To configure the FioranoMQ Server to snoop for messages, follow the steps below:

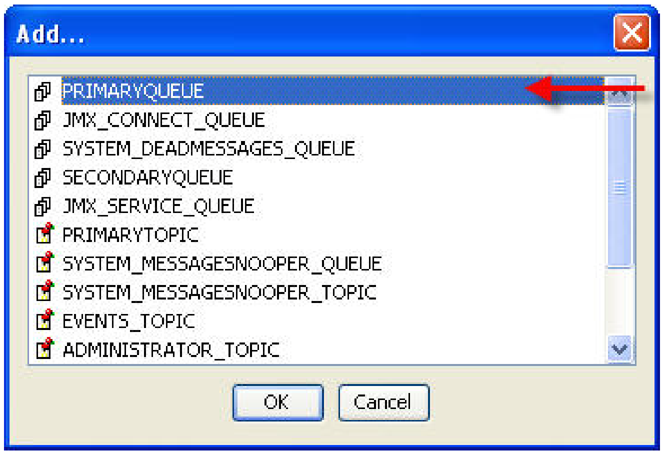
1. From the **Server Explorer** pane, select the **Snooper** node. Select either the **Queues** or the **Topics** sub-node.
2. Right-click and select the **Add Destinations in Snooper** option.



3. The Destinations in Snooper dialog box is displayed.



4. Click the **Add** button to display the **Add** dialog box. Select the destination and Click the **OK** button.



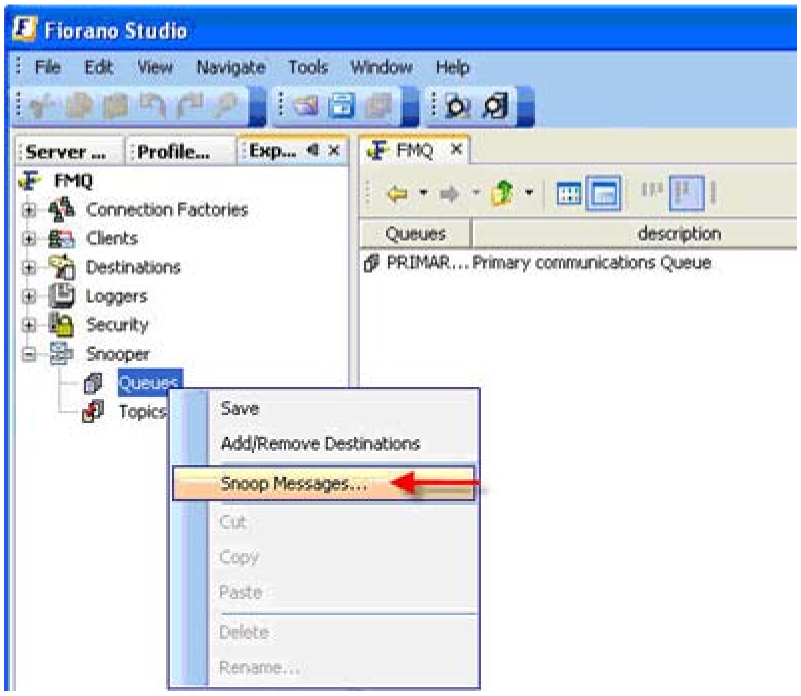
5. Click the **OK** button in the **Destinations in Snooper** dialog box to register the selected destinations.
6. All changes made using Snooper node are transient and only valid for that particular execution of the server.

To render the changes stable, click the **Save** button present on the Main toolbar to save the configuration settings in the FioranoMQ Server. If these settings are not saved, they will be lost when the FioranoMQ Server is restarted.

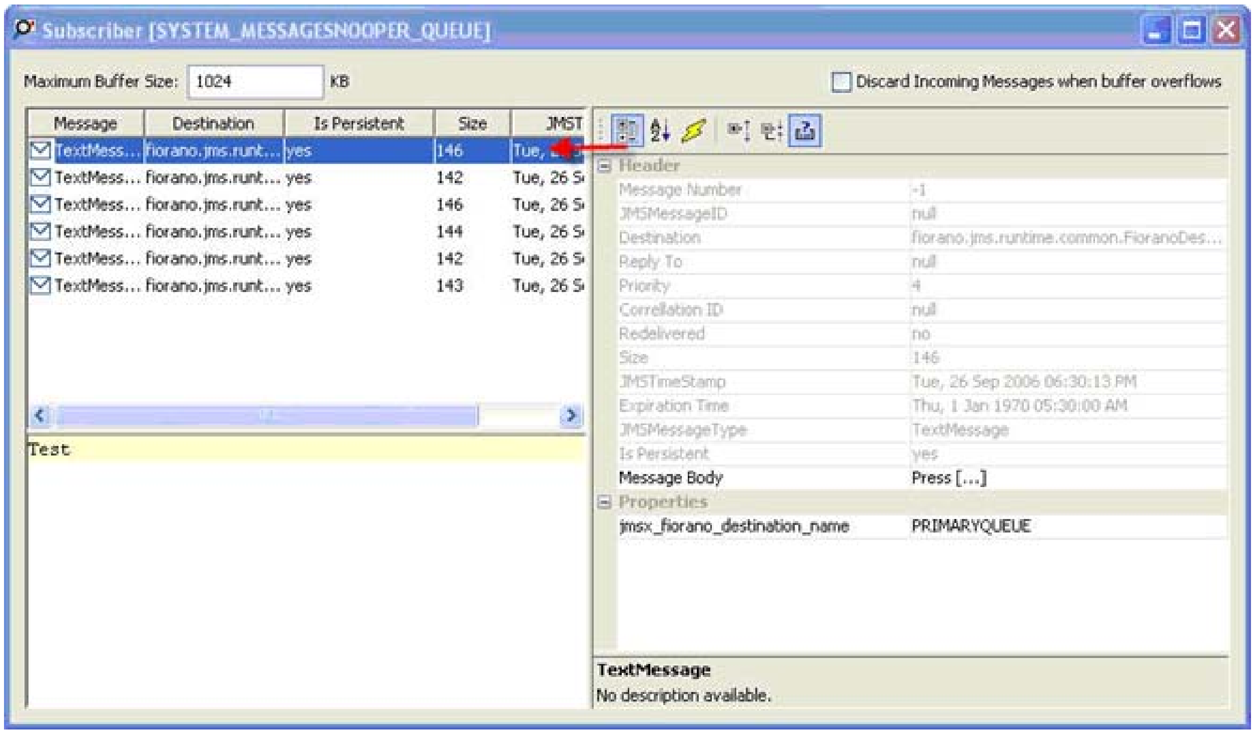
25.6.2 Snooping Messages

Snooping Messages allows the administrator to view the total number of messages, purge all the messages and refresh the browser display. Follow the steps below to browse a queue or topic:

1. From the **Snooper** node, select either the **Queues** or the **Topics** sub-node.
2. Right-click and select **Snoop Messages** from the pop-up menu.



The Snooper Window is displayed showing the snooped messages.



Subscriber [SYSTEM_MESSAGESNOOPER_QUEUE]

Maximum Buffer Size: 1024 KB

Discard Incoming Messages when buffer overflows

Message	Destination	Is Persistent	Size	JMST
TextMess...	fiorano.jms.runt...	yes	146	Tue,
TextMess...	fiorano.jms.runt...	yes	142	Tue, 26 S
TextMess...	fiorano.jms.runt...	yes	146	Tue, 26 S
TextMess...	fiorano.jms.runt...	yes	144	Tue, 26 S
TextMess...	fiorano.jms.runt...	yes	142	Tue, 26 S
TextMess...	fiorano.jms.runt...	yes	143	Tue, 26 S

Test

Header	
Message Number	-1
JMSMessageID	null
Destination	fiorano.jms.runtime.common.FioranoDes...
Reply To	null
Priority	4
Correlation ID	null
Redelivered	no
Size	146
JMSTimeStamp	Tue, 26 Sep 2006 06:30:13 PM
Expiration Time	Thu, 1 Jan 1970 05:30:00 AM
JMSMessageType	TextMessage
Is Persistent	yes
Message Body	Press [...]
Properties	
jmsx_fiorano_destination_name	PRIMARYQUEUE

TextMessage
No description available.

25.6.3 Refreshing and Saving Snooper

Using the Admin studio, it is possible to refresh the message display list and save the changes made to Snooper.

25.6.4 Refreshing Snooper

Follow the steps below to refresh the message display list:

1. Select the **Snooper** node from the **Server Explorer** pane.
2. Right-click and select the **Refresh** option from the pop-up menu.

25.6.5 Saving Snooper

Using the feature, described below, of Admin Studio it is possible to save configuration settings set to snoop messages. Saving a Snooper is particularly required in cases where the destinations are added to the snooper. Follow the steps below to save the Snooper:

1. Select the **Snooper** node from the **Server Explorer**.
2. Click the **Save** button in the Main toolbar. Alternatively, right-click and select **Save** from the pop-up menu.

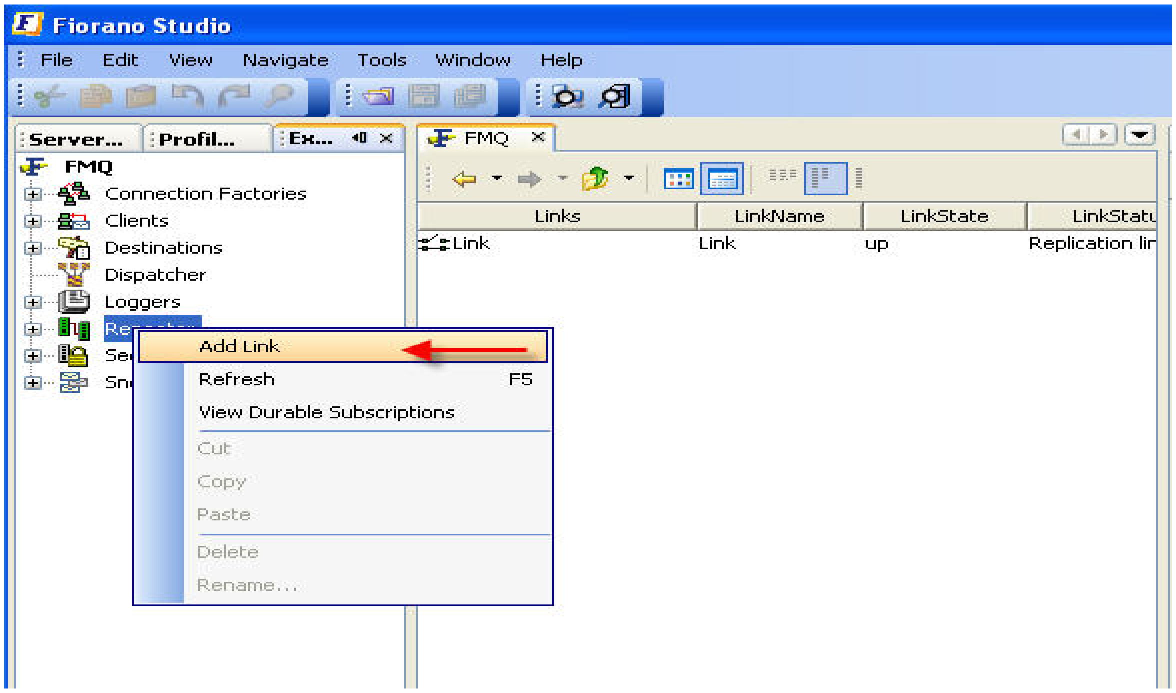
25.7 Working with Repeater

Using the **Repeater** node it is possible to keep track of the repeaters in the network. In addition, the **Repeater** node makes it possible to administer various repeaters across the network. The following sub-sections explain, in detail, the various operations that can be undertaken to manage a Repeater. This module is visible only after the repeater is configured in the server.

25.7.1 Adding a Link

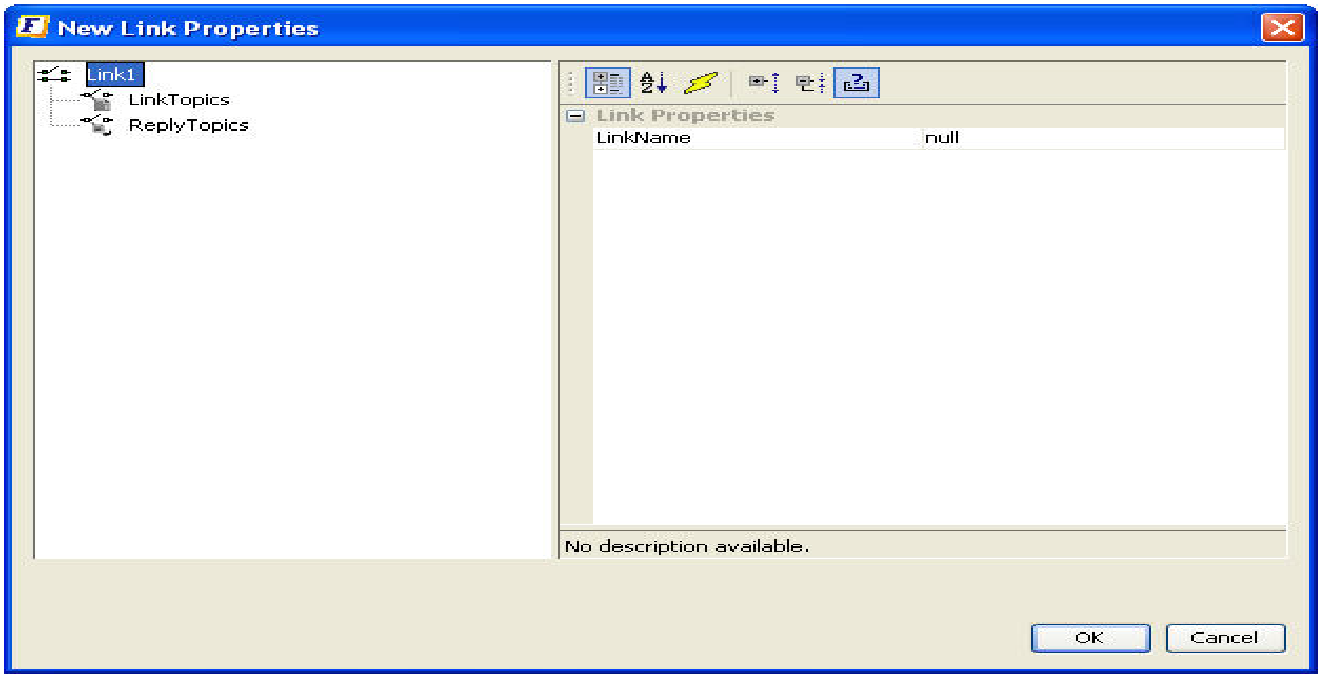
The administrator can create new replication links dynamically. This enables the applications to replicate messages on topics that are created after the repeater has started. Information such as the new link name, the source and target servers between which the link needs to be created, the protocol to be used by the connection and login information need to be provided when a new link is added to the repeater. To add a new link, follow the steps below:

1. Select the **Repeater** node from the **Server Explorer** pane.
2. Right-click and select the **Add Link** option from the pop-up menu.



Links	LinkName	LinkState	LinkStatu
Link	Link	up	Replication lir

3. The **New Link Properties** dialog box is displayed.



LinkName	null

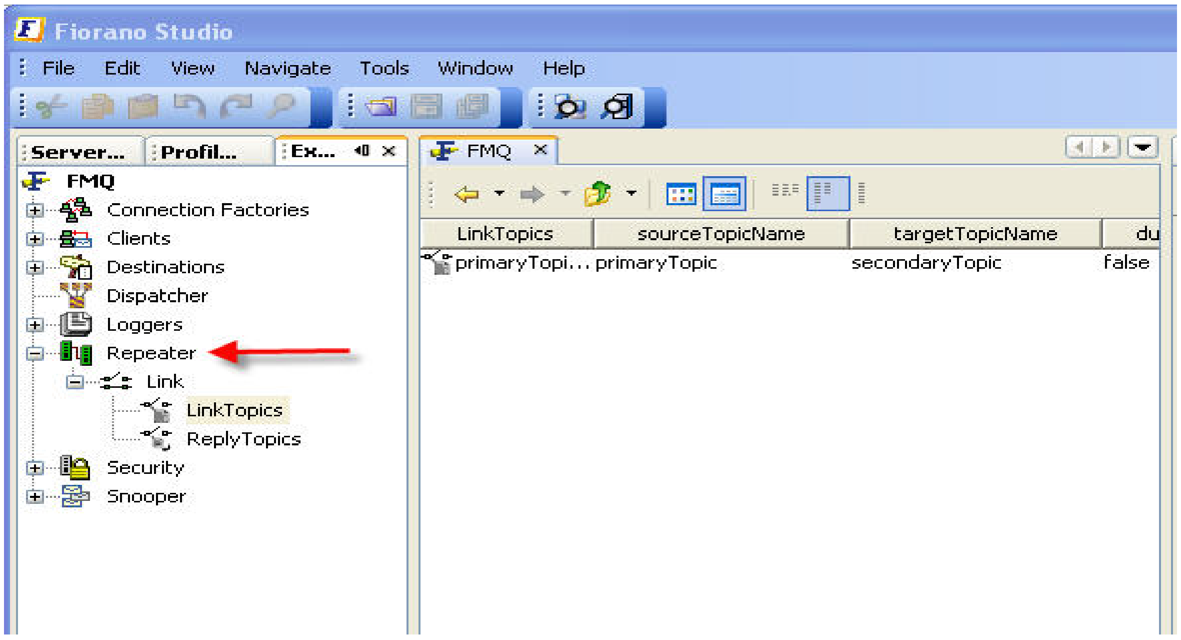
No description available.

4. Specify the link properties and Click the **OK** button. The new link is added as a node and shown in the **Server Explorer** pane.

25.7.2 Adding a Link Topic

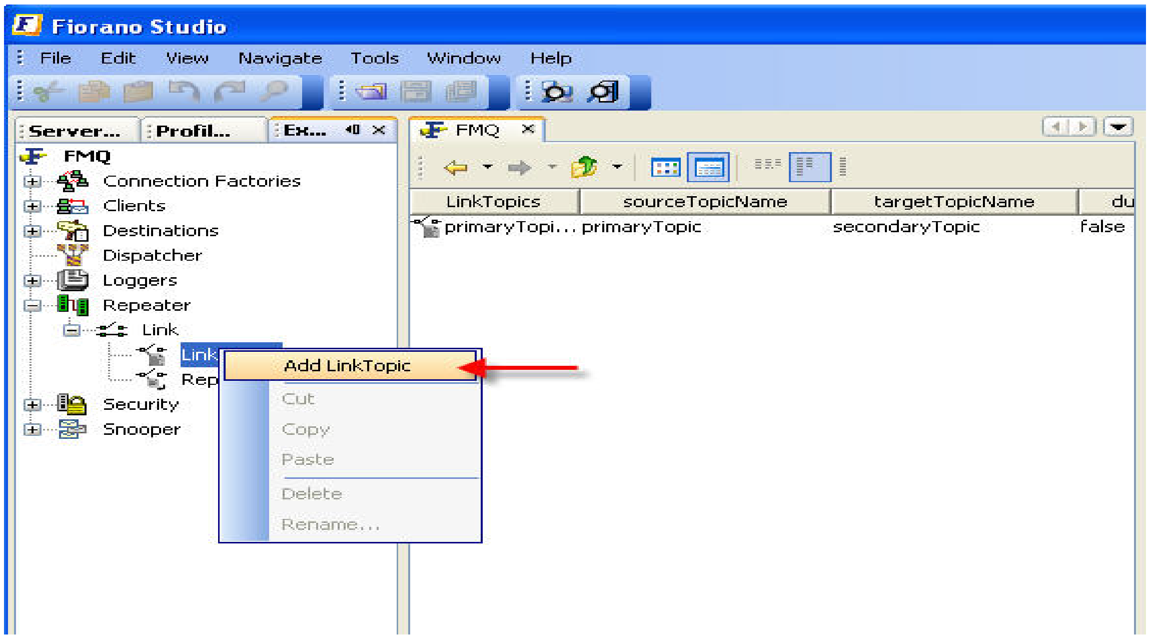
The administrator can add one or more link topics to an existing topic. While doing so, the administrator must specify information such as the source topic name, the target topic name and the message selector. To add a link topic, follow the steps below:

1. Expand the **Link** sub-node from the **Repeater** node in the **Server Explorer** pane. The various link topics configured within the repeater are displayed.



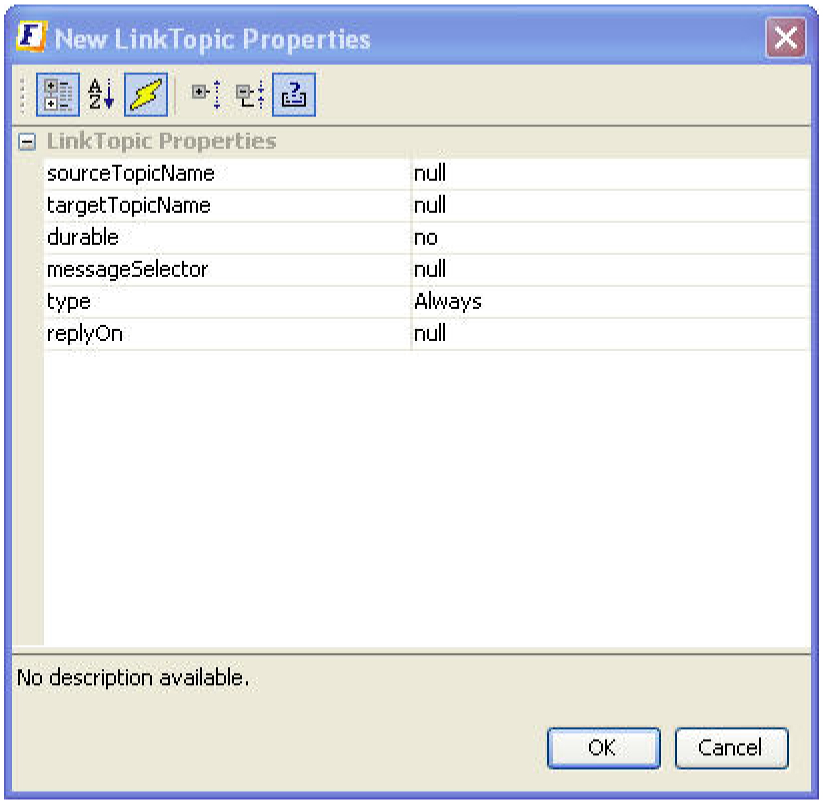
LinkTopics	sourceTopicName	targetTopicName	du
primaryTopi...	primaryTopic	secondaryTopic	false

2. Right-click any of the selected **Link Topics** and select the **Add LinkTopic** option from the pop-up menu.



LinkTopics	sourceTopicName	targetTopicName	du
primaryTopi...	primaryTopic	secondaryTopic	false

3. The **New LinkTopic Properties** dialog box is displayed.



sourceTopicName	null
targetTopicName	null
durable	no
messageSelector	null
type	Always
replyOn	null

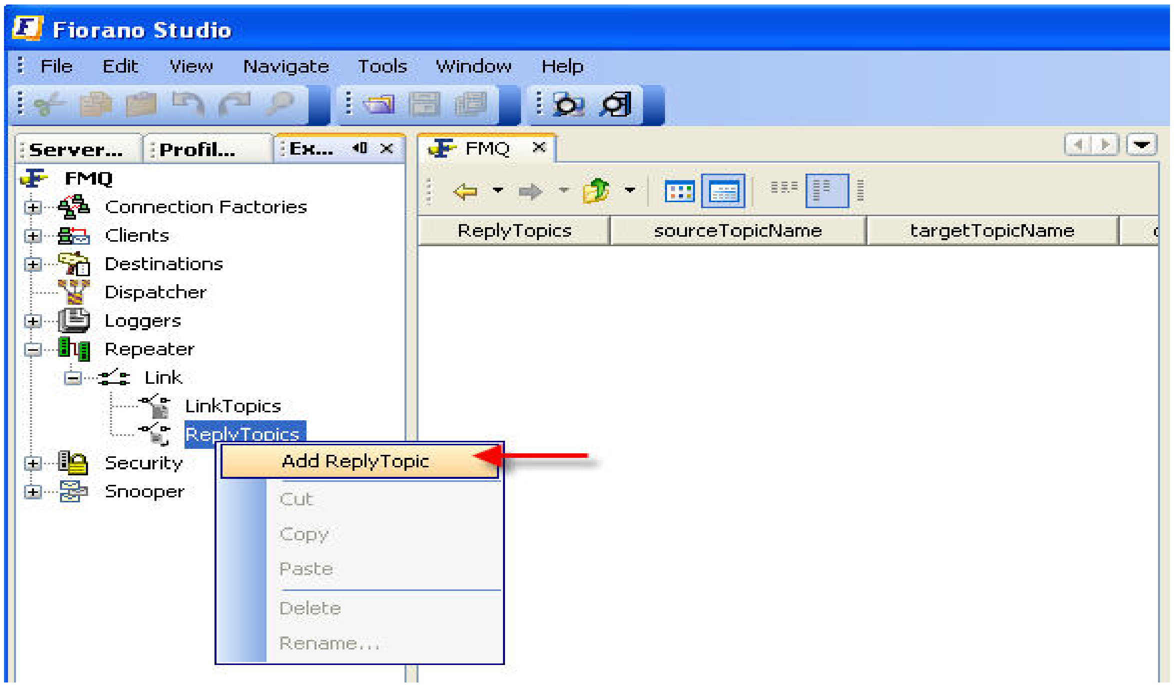
No description available.

4. Specify the properties and Click the **OK** button. The new link topic is added as a node and shown in the **Server Explorer** pane.

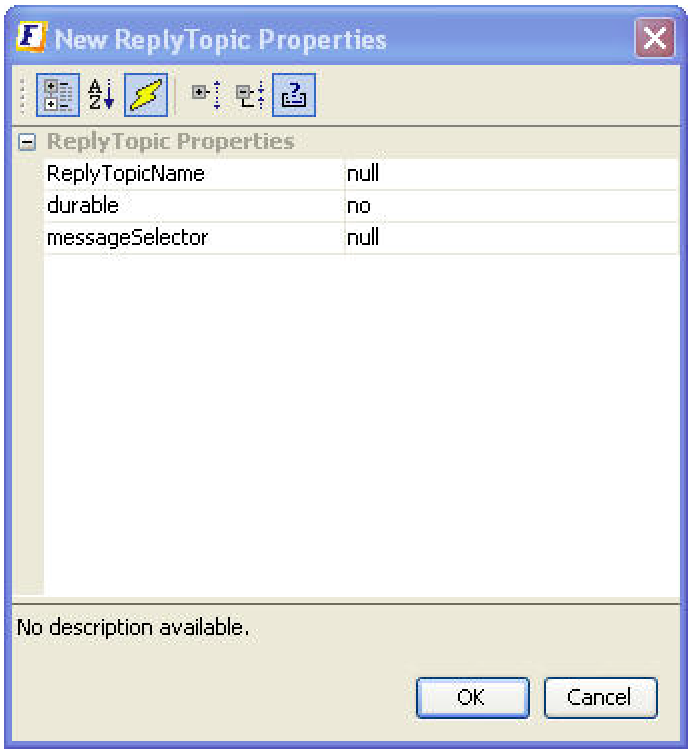
25.7.3 Adding a Reply Topic

The administrator can add one or more reply topics to an existing topic. While doing so, the administrator must specify information such as the source topic name, the target topic name and message selector. To add a Reply topic, follow the steps below:

1. Expand the **Link** sub-node from the **Repeater** node in the **Server Explorer** pane. Reply topics configured within the repeater are displayed.
2. Right-click any of the selected **Topics** and select the **Add ReplyTopic** option from the pop-up menu.



3. The **New LinkTopic Properties** dialog box is displayed. Specify the properties and click the **OK** button. The new Reply Topic is added as a node and shown in the **Server Explorer** pane.



ReplyTopicName	null
durable	no
messageSelector	null

No description available.

25.7.4 Removing a Link

The administrator can remove a selected link from the repeater by following the steps below:

1. Select the **link** to be removed from the **Server Explorer** pane.
2. Right-click and select **Delete** from the pop-up menu.
3. The **Confirm Object Deletion** dialog box is displayed. Click the **Yes** button. The selected link is removed from the cluster.

25.7.5 Removing a Link Topic

The administrator can remove selected link topics from the repeater by following the steps below:

1. Select the **link** to be removed from the **Server Explorer** pane.
2. Right-click and select **Delete** from the pop-up menu.
3. The **Confirm Object Deletion** dialog box is displayed. Click the **Yes** button. The selected Link Topic is deleted from the cluster.

25.7.6 Viewing Durable Subscribers for a Repeater

Durable subscribers can be viewed by right-clicking the repeater and selecting the **View Durable Subscribers** option from the pop-up menu. A list of all durable subscribers is displayed in the **Details** pane.

25.7.7 Refreshing Repeater

Using the Admin studio it is possible to allow refreshing the status of repeaters in a cluster.

Follow the steps below to refresh the Repeater:

1. Select the **Repeater** node from the **Server Explorer** pane.
2. Right-click and select **Refresh** from the pop-up menu. Alternatively, press the F5 key to refresh.

25.8 Working with Dispatcher

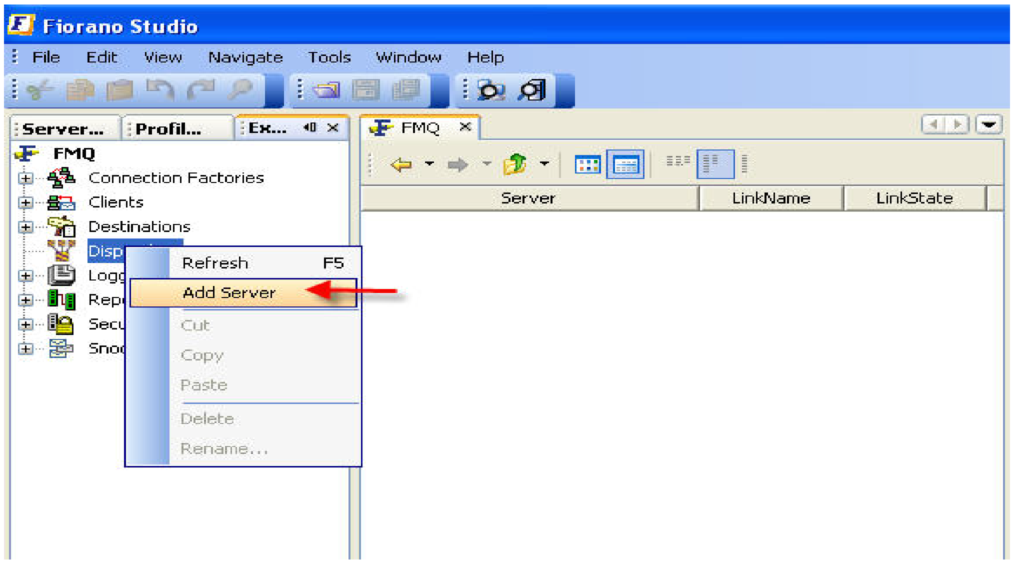
The **Dispatcher** node displays the status of the FioranoMQ cluster. In addition, it is possible to add or remove an MQ Server from a cluster, as well as to change the Preferred Server Status using the **Dispatcher** node.

25.8.1 Adding a Server

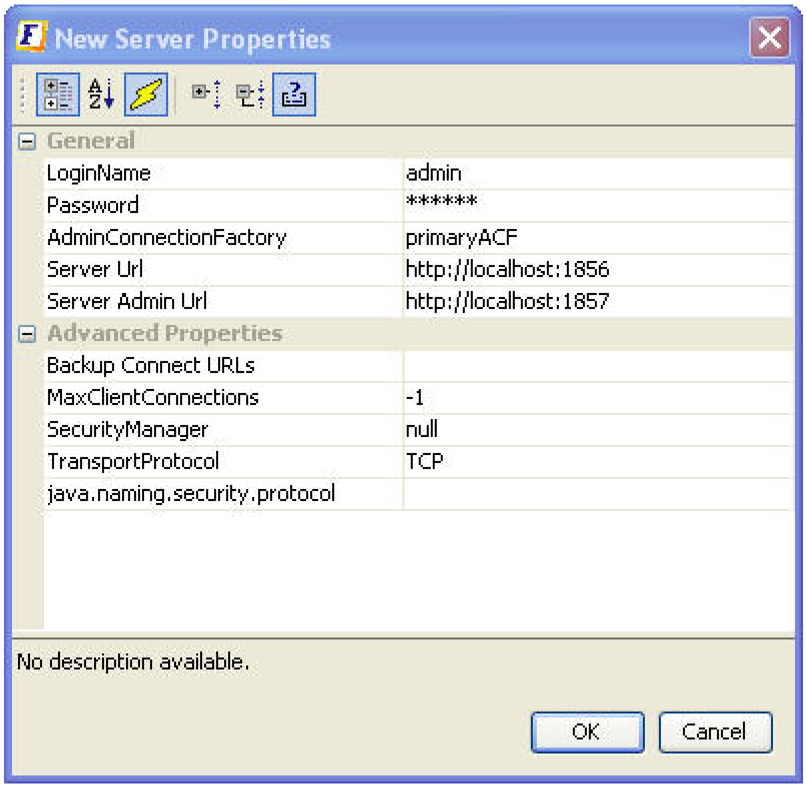
The administrator can add a new Server to the dispatcher. To add a new Server to the dispatcher, follow the steps below:

1. Select the **Dispatcher** node from the **Server Explorer** pane.

2. Right-click and select the **Add Server** option from the pop-up menu.



3. The **New Server Properties** dialog box is dispalyed. The Server can have two types of properties: **General** and **Advanced**. Configure the Server as per requirements.



General	
LoginName	admin
Password	******
AdminConnectionFactory	primaryACF
Server Url	http://localhost:1856
Server Admin Url	http://localhost:1857
Advanced Properties	
Backup Connect URLs	
MaxClientConnections	-1
SecurityManager	null
TransportProtocol	TCP
java.naming.security.protocol	

No description available.

4. Specify the Server properties and Click the **OK** button. The new Server is added as a node.

25.8.2 Removing a Server

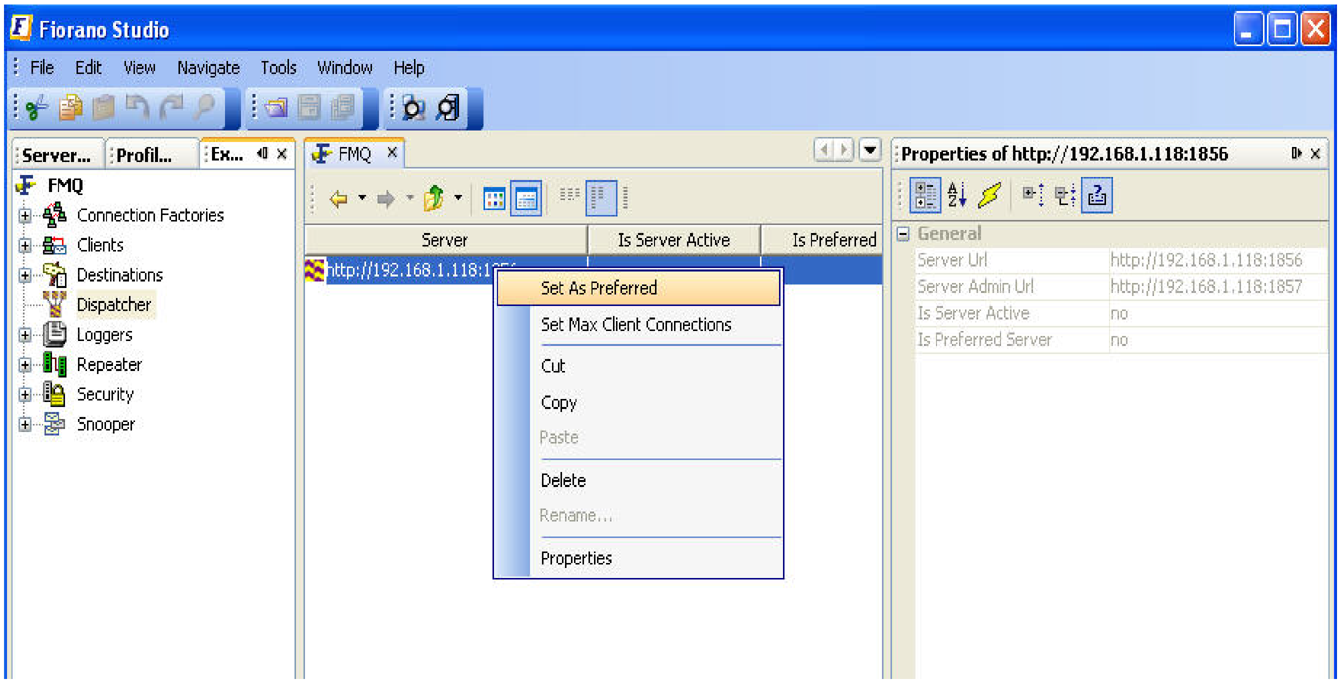
The administrator can remove the selected Server from the cluster by following the steps below:

1. Select the **name of the server** to be removed from the **Server Explorer** pane.
2. Right-click and select **Delete** from the pop-up menu.
3. The **Confirm Object Deletion** dialog box is displayed. Click the **Yes** button. The selected server is deleted and is no longer visible in the cluster.

25.8.3 Setting the Preferred Server

Applications can instruct the dispatcher to connect to the preferred Server of a cluster, bypassing the load balancing mechanism. This function can be used by any application that always needs to connect to one particular Server in the cluster. The administrator can set a Server to be the preferred server by following the steps below:

1. Select the **name of the server** that is to be set as the preferred Server from the **Server Explorer** pane.
2. Right-click and select **Set as Preferred** from the pop-up menu.

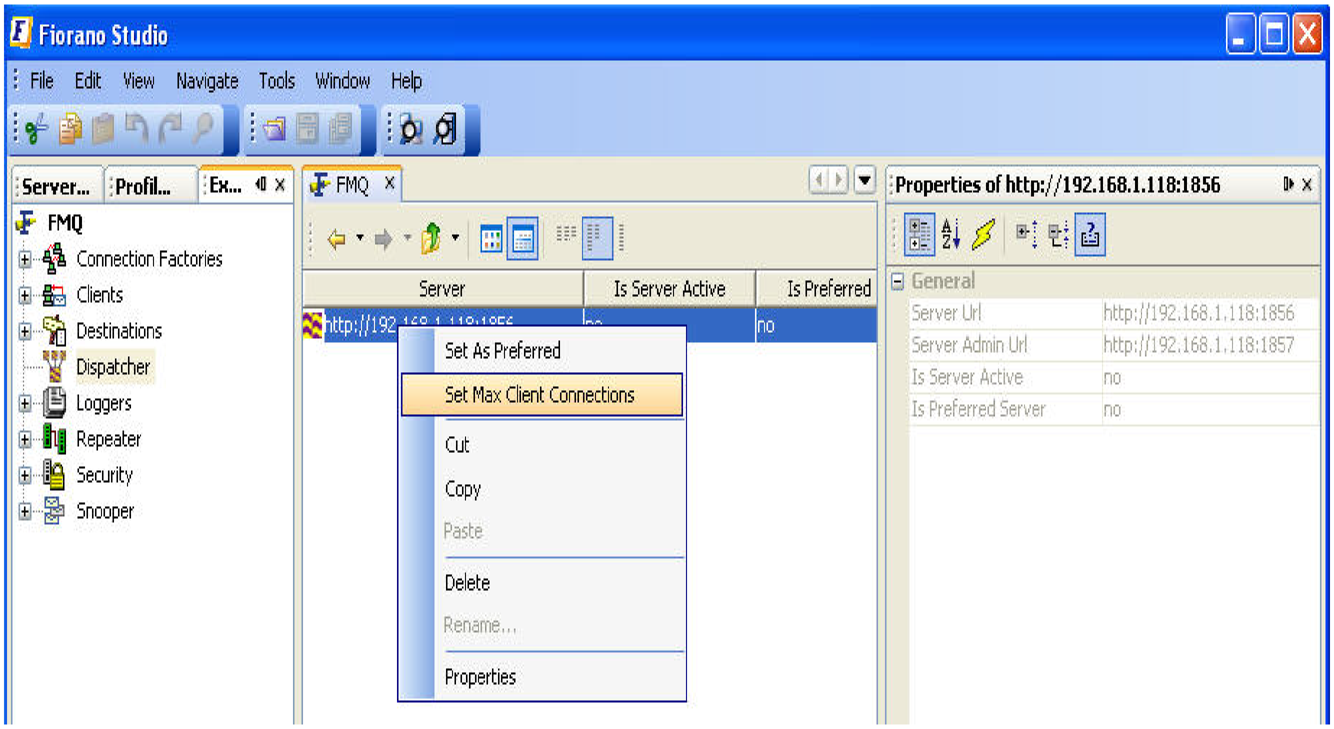


Server Url	http://192.168.1.118:1856
Server Admin Url	http://192.168.1.118:1857
Is Server Active	no
Is Preferred Server	no

25.8.4 Setting Number of Client Connections

The administrator has the right to set the maximum number of client connections that can be created on a member Server of the cluster. This number is used to compute the least loaded server in the cluster. Follow the steps below to set the maximum number of client connections:

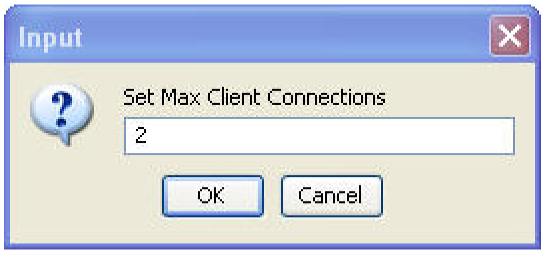
1. Select the **server name** from the **Server Explorer** pane.
2. Right-click and select the **Set Max Client Connections** option from the pop-up menu. The **Input** dialog box is displayed.



Server	Is Server Active	Is Preferred
http://192.168.1.118:1856	no	no

General	
Server Url	http://192.168.1.118:1856
Server Admin Url	http://192.168.1.118:1857
Is Server Active	no
Is Preferred Server	no

3. Provide the number in the **Set Max Client Connections** box and Click the **OK** button



Set Max Client Connections: 2

25.8.5 Refreshing Dispatcher

Using the Admin Studio it is possible to refresh the status of servers in the cluster. Follow the steps below to refresh the dispatcher:

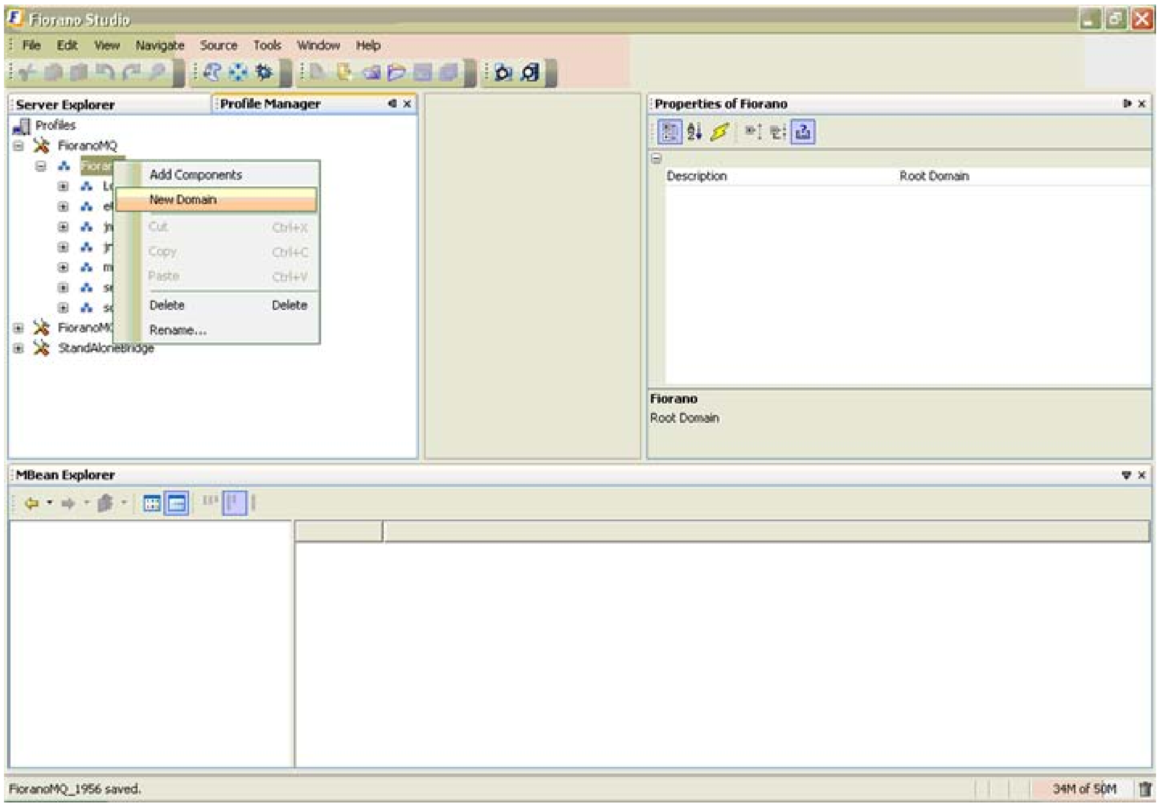
1. Select the **Dispatcher** node from the **Server Explorer** pane.
2. Right-click and select **Refresh** from the pop-up menu. Alternatively, press the **F5** key to refresh.

25.9 Working with Bridge

The Bridges node in the Fiorano Admin Studio makes it possible to manage all the bridges that connect the FioranoMQ Server to another MQ server or any other messaging Server. The admin tool enables adding links, adding channels to a link and monitoring the status of every bridge running within the cluster. This module is visible only after the bridge is configured in the Server.

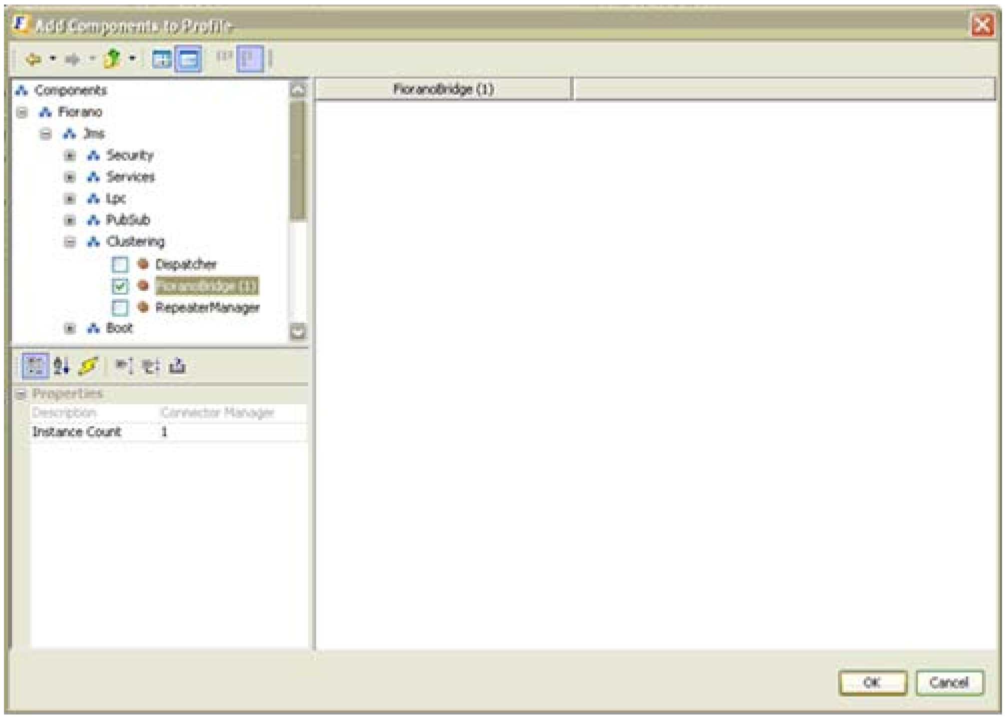
25.9.1 Adding the Bridge to the FioranoMQ Profile

1. In Studio open ProfileManager and open the FioranoMQ profile. Right-click **Fiorano** and select **New Domain** as shown below.



2. Name the Domain as **Bridge**, for User convenience.

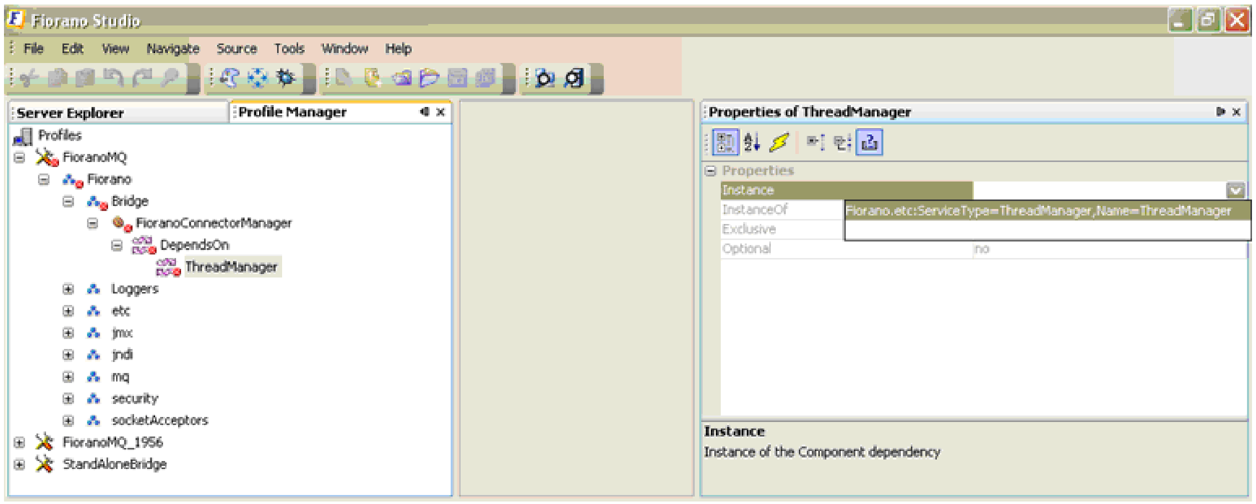
3. Right-click on **Bridge** (new domain that was added in the previous step) and click on **Add Components**. Go to Fiorano→Jms→Clustering and select FioranoBridge. Click **ok**. The FioranoConnectorManager is added to the Bridge.



Properties

Description	Connector Manager
Instance Count	1

4. Navigate to **FioranoConnectorManager→DependsOn→ThreadManager**. By clicking on the **ThreadManager** the User can see properties on the right-hand side of studio. By default, instance property will be null. Select the only instance present there. Name the **FioranoConnectorManager** in properties.

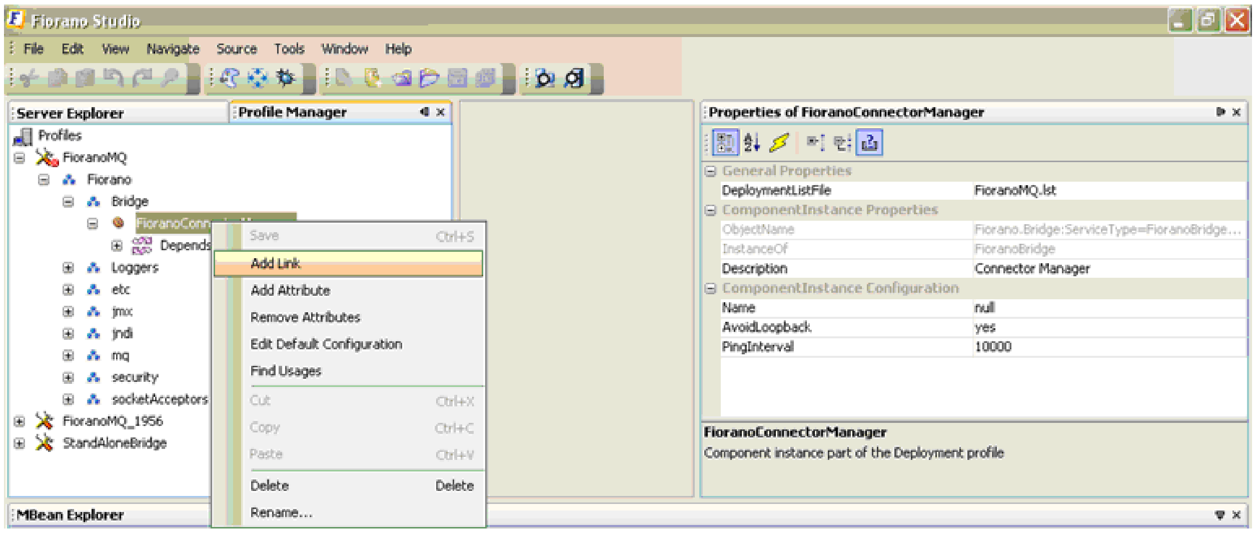


Properties

Instance	
InstanceOf	Fiorano.etc:ServiceType=ThreadManager,Name=ThreadManager
Exclusive	
Optional	no

Instance
Instance of the Component dependency

5. Right-click on **FioranoConnectorManager** and **addLink** as shown below. Name the link in properties.



General Properties

DeploymentListFile	FioranoMQ.lst

ComponentInstance Properties

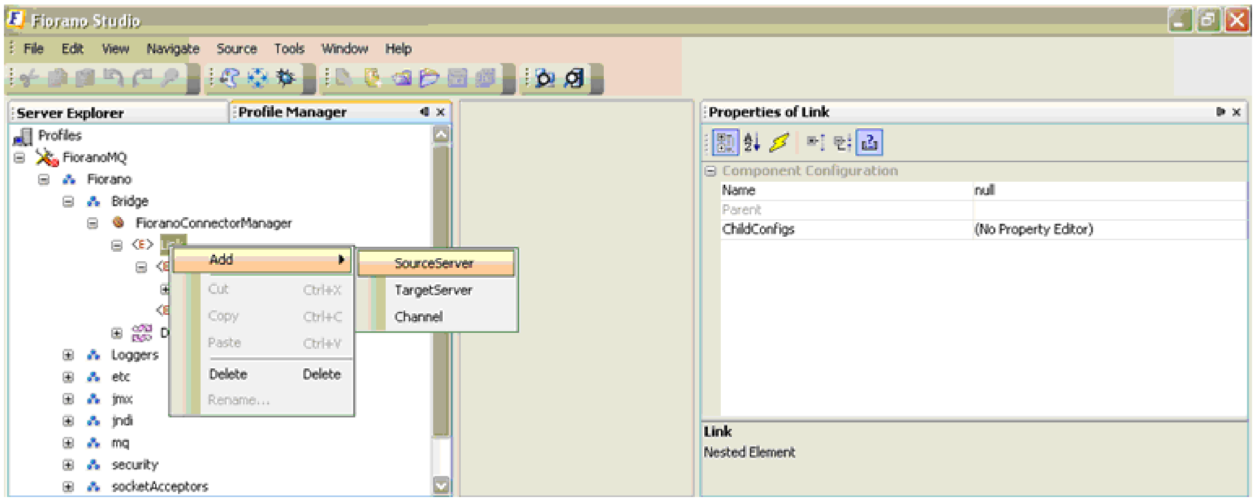
ObjectName	Fiorano.Bridge:ServiceType=FioranoBridge...
InstanceOf	FioranoBridge
Description	Connector Manager

ComponentInstance Configuration

Name	null
AvoidLoopback	yes
PingInterval	10000

FioranoConnectorManager
Component instance part of the Deployment profile

6. Right-click on **Link** and add source Server, target Server and channel. Add **connectionInfo** to the source Server and to the target Server. Add source queue and target queue to the channel. Add queueInfo to source queue and target queue. All components added need to be configured. Configuration techniques are available in Chapter: 12 Bridge.



Component Configuration

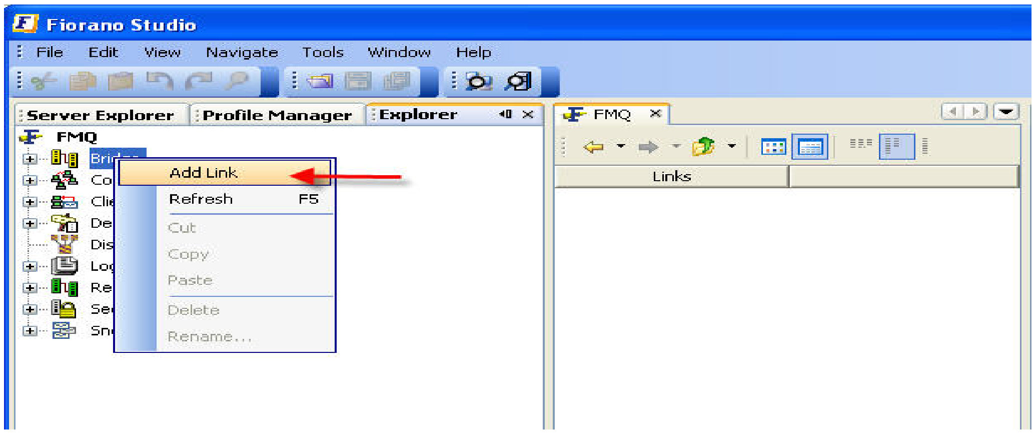
Name	null
Parent	
ChildConfigs	(No Property Editor)

Link
Nested Element

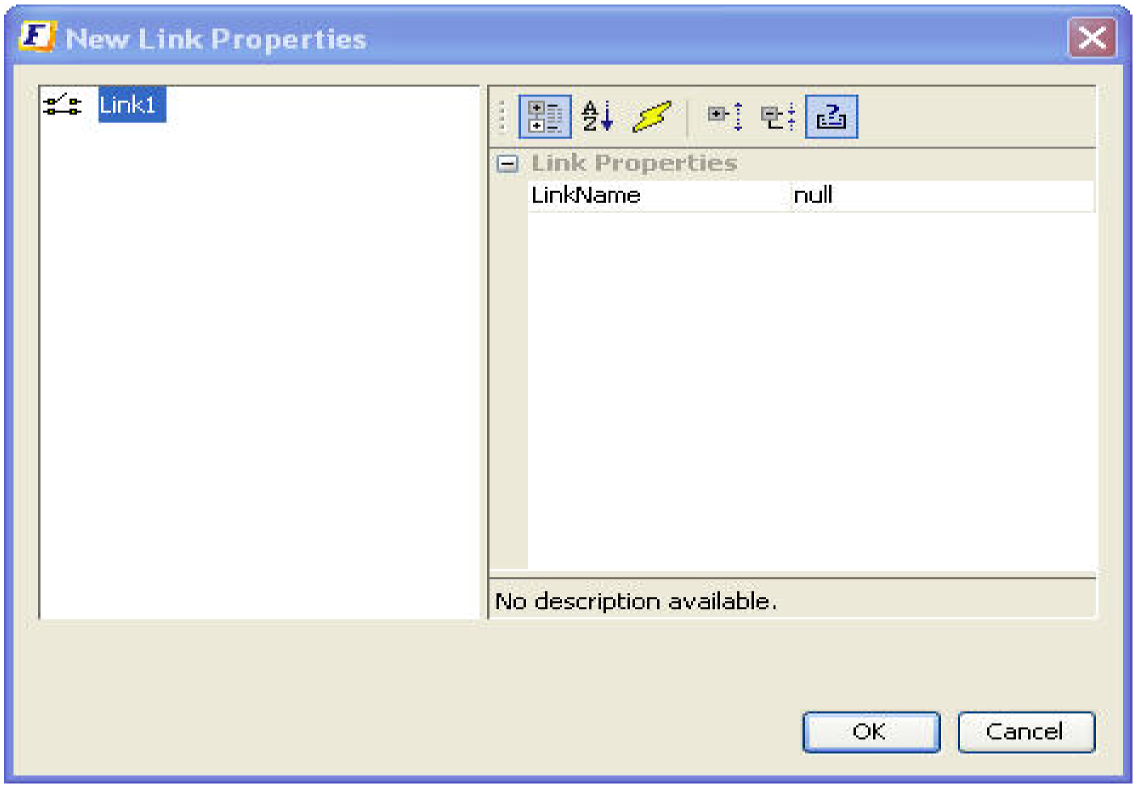
25.9.2 Adding a Link

The administrator can create new links dynamically in a Bridge. This enables the application to send and receive messages on queues created after the bridge has started. A link can contain one or more channels. Information such as the link name, the source and target Servers between which the link needs to be created, the protocol to be used by the connection and login information need to be provided when a new link is added to the Bridge. To add a new link, follow the steps below:

1. Select the **Bridge** node from the **Server Explorer** pane.
2. Right-click and select the **Add Link** option from the pop-up menu.



3. The **New Link Properties** dialog box is displayed.



Link Properties

LinkName null

No description available.

4. Specify the link properties and click the **OK** button. The new link is added as a node and shown in the **Server Explorer** pane.

25.9.3 Removing a Link

The administrator can remove the selected link from the repeater by following the steps below:

1. Select the **link** to be removed from the **Server Explorer** pane.
2. Right-click and select **Delete** from the pop-up menu.

3. The **Confirm Object Deletion** dialog box is displayed. Click the **Yes** button. The selected link is deleted and is no longer visible in the cluster.

25.9.4 Starting a Link

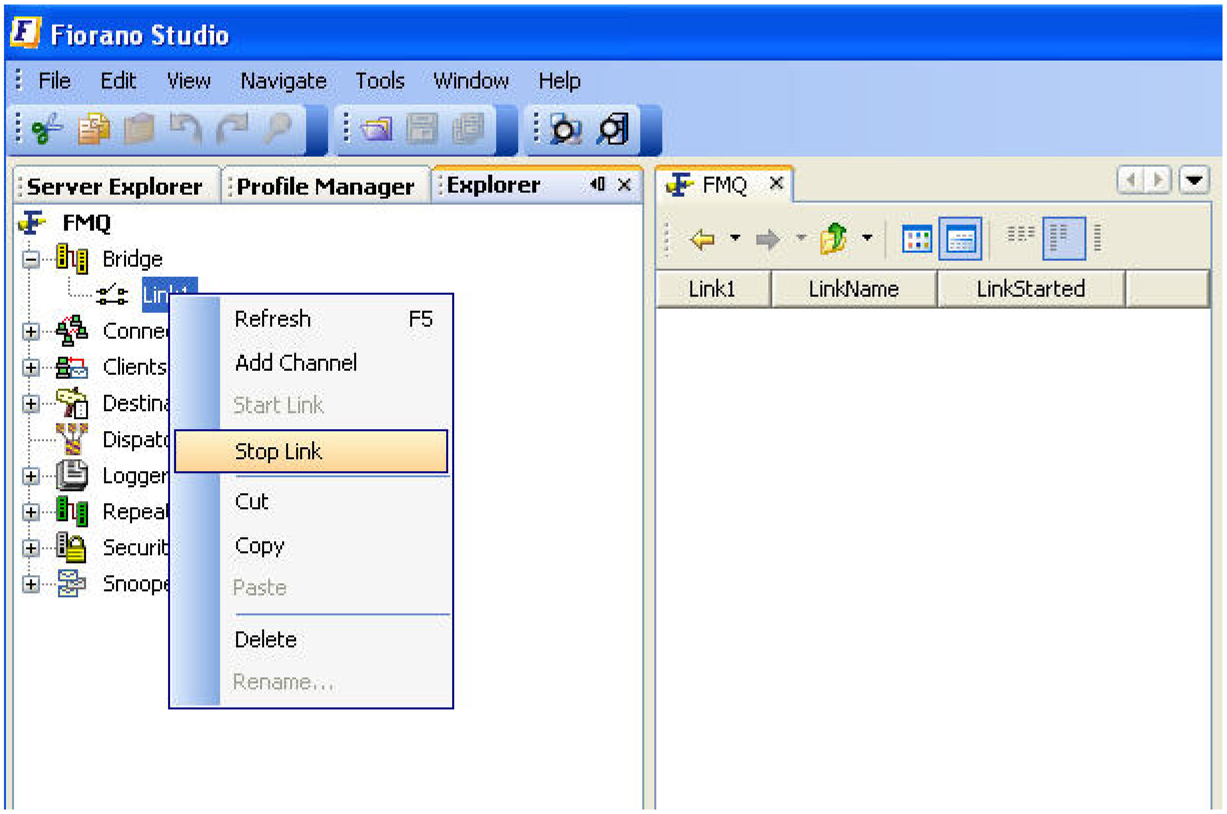
To start a link:

Select the **link** from the Bridge in the **Server Explorer** pane. Right-click and select the **Start Link** option from the pop-up menu.

25.9.5 Stopping a Link

To stop a link:

Select the **Link** from the Bridge in the **Server Explorer** pane. Right-click the link and select the **Stop Link** option from the pop-up menu.

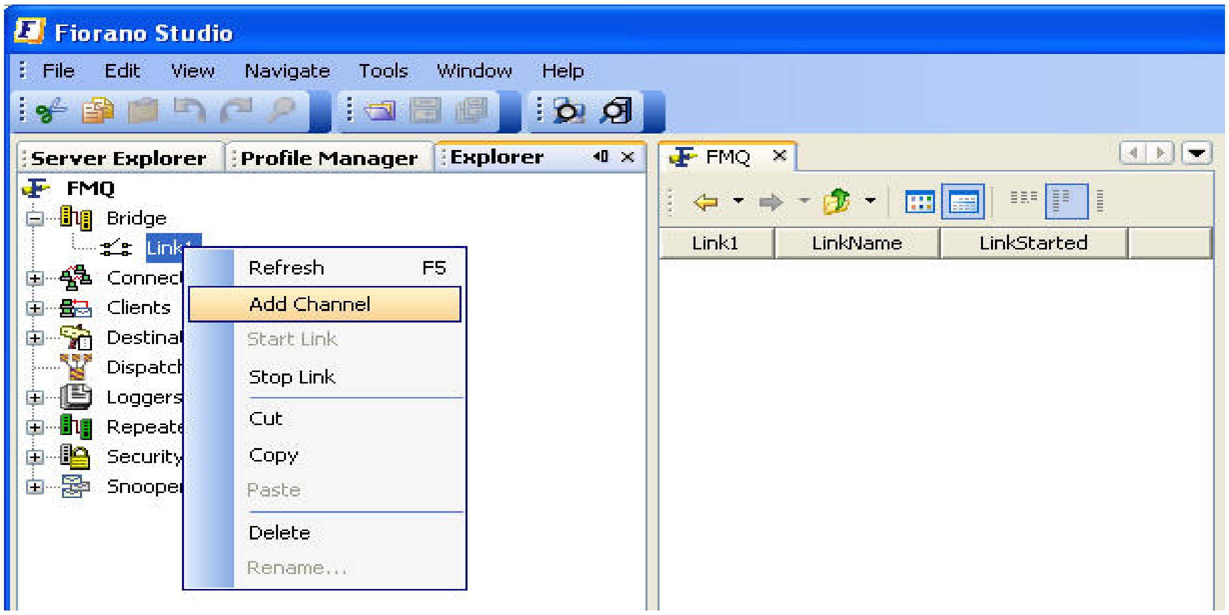


25.9.6 Adding a Channel to a Link

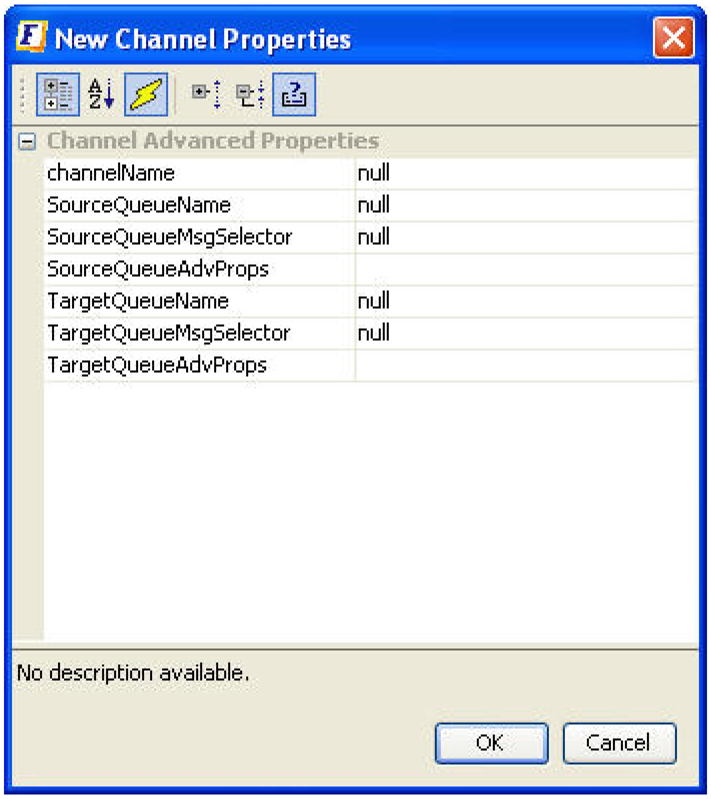
The administrator can add channels to a link dynamically. To add a channel to a link the administrator must provide information such as source and target queues. A link can have multiple Channels mapped to different Queues.

1. Expand the **Link** sub-node from the **Bridge** node in the **Server Explorer** pane. The various channels configured within the bridge are displayed.

2. Right-click any of the **Channels** and select the **Add Channel** option from the pop-up menu.



3. The **New Channel Properties** dialog box is displayed.



channelName	null
SourceQueueName	null
SourceQueueMsgSelector	null
SourceQueueAdvProps	
TargetQueueName	null
TargetQueueMsgSelector	null
TargetQueueAdvProps	

No description available.

4. Specify the properties and Click the **OK** button. The new channel is added as a node and shown in the **Server Explorer** pane.

25.9.7 Removing a Channel from a Link

The administrator can remove a channel from a link on a Bridge by following the steps below:

1. Select the **channel** to be removed from the **Server Explorer** pane.
2. Right-click and select **Delete** from the pop-up menu.
3. The **Confirm Object Deletion** dialog box is displayed. Click the **Yes** button. The selected channel is deleted and is no longer visible in the cluster.

Chapter 26: Performance Tuning And Deployment Parameters

This chapter explains the various parameters that should be modified to obtain the best performance from FioranoMQ. This chapter also provides information on deploying FioranoMQ and additional software components provided with FioranoMQ in addressing various scaling and clustering issues.

26.1 Performance Tuning Parameters

The default configuration of FioranoMQ provides optimum performance under most traffic loads and for most common message sizes (ranging from 0-10 KB).

FioranoMQ administrators can tune the configurations, below, to optimize the performance of the FioranoMQ Server.

26.1.1 PTP Configuration Parameters

FioranoMQ provides the configuration parameters, below, to tune the point-to-point messaging model:

26.1.1.1 In-Memory Persistent Message Buffer

FioranoMQ server buffers the persistent messages in its in-memory cache besides storing these messages in the persistent store. This buffering is done to ensure fast delivery of messages to the listening receivers. The variable named **PersistentInMemoryBufferSize** depicts the size of the in-memory buffer whose default value is 512KB (512*1024). This value can be increased when the message send rate is high. This is because when the send rate is high, the buffer gets filled up faster than when the send rate of messages is lower.

The steps below enable the configuration of the In-Memory Persistent Message Buffer parameter using the Fiorano Admin Studio in online mode:

1. Invoke the Fiorano Admin Studio and login to the FioranoMQ Server through **JMX** login.
2. Select **Fiorano** > **mq** > **ptp** > **PtPManager** > **QueueingSubSystem** > **config** from the Server Explorer pane.
3. In the **Properties** pane, type in the new value against the property named **PersistentInMemoryBufferSize** and press Enter.

26.1.1.2 In-Memory Non-Persistent Message Buffer

The FioranoMQ Server stores the published non-persistent messages in an in-memory table. The size of this buffer is monitored by the parameter **NPInMemoryBufferSize** whose default value is 1MB. This buffer can gets full if the message send rate is quite high as compared to the message receive rate. When this buffer fills up, the FioranoMQ senders cannot push new messages into the buffer and messages remain blocked till a receiver is activated on the concerned queue to empty the buffer.

It is recommended that the size of the buffer is set to a value where chances of it becoming full are minimized, particularly in cases where the senders are sending messages at a much faster rate than the rate at which they are being received.

The steps below enable the configuration of the In-Memory Non-Persistent Message Buffer parameter through Fiorano Admin Studio in online mode:

1. Through the Fiorano Admin Studio login to the FioranoMQ Server using**JMX** login.
2. Select **Fiorano** > **mq** > **ptp** > **PtPManager** > **QueueingSubSystem** > **config** from the **Server Explorer** pane.
3. In the **Properties** pane, type in the new value of the property named **InMemoryBufferSize** and press Enter.

26.1.1.3 Prefetch Count

Prefetch count is the number of messages requested by a client from the Server in one 'receive call'. Prefetch count can be set through the **PrefetchCount** parameter using Admin Studio in the offline mode. The default value of this variable is 3. PTP prefetching enables the FioranoMQ Server to deliver a maximum number of prefetch count messages to the receiver on each receive call.

Perform the steps below to configure the Prefetch Count parameter:

1. Open the Fiorano Admin Studio.
2. Select **Tools > Configure Profile** from the menu bar. Select the **FioranoMQ** folder, and click the **Open** button. FioranoMQ is now in offline mode.
3. Navigate to **FioranoMQ -> Fiorano -> mq -> ptp -> QueuingSubSystem** in the **Profile Manager** pane. The properties of the PTP Manager are displayed in the **Properties** pane.
4. Select the **ellipsis** against the parameter **PrefetchCount**. Type in the new value in the **PrefetchCount** dialog box and click **OK**.
5. Right-click on the FioranoMQ domain in the **Server Explorer** pane and select the **Save** option from the shortcut menu.

26.1.1.4 Prefetch Threshold

Prefetch threshold denotes the minimum number of messages in the local buffer of the client that trigger a request for delivery of more messages to the client. Prefetch threshold can be set through a parameter named **PrefetchThreshold** using the Fiorano Admin Studio in the offline mode.

The default value of the Prefetch Threshold variable is 1. The number of messages that are requested from the server when the threshold limit is reached is equal to the prefetch count.

Perform the steps below to configure the Prefetch Threshold parameter:

1. Open the Fiorano Admin Studio.
2. Select **Tools > Configure Profile** from the menu bar. Select the **FioranoMQ** folder, and click the **Open** button. FioranoMQ is now in offline mode.
3. Navigate to **FioranoMQ -> Fiorano -> mq -> ptp -> QueuingSubSystem** in the **Profile Manager** pane. The properties of the PTP Manager are displayed in the **Properties** pane.
4. Select the **ellipsis** against the parameter **PrefetchThreshold**. Type in the new value in the **PrefetchThreshold** dialog box and click **OK**.
5. Right-click on the FioranoMQ domain in the **Profile Manager** pane and select the **Save** option from the shortcut menu.

26.1.1.5 PTP Prefetch size

The PTP Prefetch size parameter is used to specify the total size of messages (in bytes) that will be sent by the FioranoMQ Server on each receive call made by the client. Prefetch size can be set through a parameter named **MaxPrefetchSize** using the Fiorano Admin Studio in the offline mode. The default value of this variable is 512 KB. If both Prefetch Size and Prefetch Count are specified,the number of messages sent by FioranoMQ server is controlled by the minimum value among these two parameters.

Perform steps below to configure the Prefetch Size parameter:

1. Open Fiorano Admin Studio.
2. Select **Tools > Configure Profile** from the menu bar. Select the **FioranoMQ** folder and click the **Open** button. FioranoMQ is now in offline mode.
3. Navigate to **FioranoMQ -> Fiorano -> mq -> ptp -> QueuingSubSystem** in the **Profile Manager** pane. The properties of the PTP Manager are displayed in the **Properties** pane.
4. Select the **ellipsis** against the parameter **MaxPrefetchSize**. Type in the new value in the **MaxPrefetchSize** dialog box and click **OK**.
5. Right-click on the FioranoMQ domain in the **Profile Manager** pane and select the **Save** option from the shortcut menu.

26.1.1.6 Queue Sender Blocking Interval

This parameter represents the period, in milliseconds, for which the sender will be blocked (while it is unable to push more messages in the In-Memory Buffer maintained by the FioranoMQ Server). The parameter **SenderBlockingInterval** can be set through the Fiorano Admin Studio in online mode. The default value of this variable is 10 milliseconds. The Sender Blocking Interval parameter is used only where persistent messages are present. Setting this parameter to -1 allows a sender to continuously publish messages without blocking them.The Sender Blocking Interval is ignored if no receivers are registered on a queue.

Perform the steps below to configure the Sender Blocking Interval parameter:

1. Open Fiorano Admin Studio and login to the FioranoMQ server.
2. Select **Destinations > Queues** sub-node from the **Server Explorer** pane.
3. Select the Queue name whose properties are to be displayed.
4. In the **Properties** pane, type in the new value of **SenderBlockingInterval** and press Enter.

26.1.1.7 Queue Behavior On Buffer Overflow

It determines the Queue behavior for the incoming messages when the size of the in-memory buffer (InMemoryBufferSize) used for storing NON_PERSISTENT messages reaches its maximum limit.

If push operation has unsuccessfully tried 'MaxPushAttempts' times to push the message into the in-memory buffer (because of inMemoryBufferSize exceeded), three strategies can be followed:

1. Throwing exception
2. Drop oldest message and push the new message
3. Store the new message into disk

Throwing exception (Exception):

This option will throw back an exception indicating in-memoryBuffer is full back to the client.

Drop oldest message and push the new message (DropOldestMessage):

This option will drop the old message present in the in-memoryBuffer and will give place for the new message.

Store the new message into disk (WriteToDisk):

This option will store NP messages (only the messages that arrives when the in-memoryBuffer is full) in the disk. In case of HA_replicated, the NP messages stored in the disk will NOT be replicated to the secondary server.

26.2 PubSub Configuration Parameters

FioranoMQ offers a very high message throughput and low latency for non-persistent messages. The boost in delivery rates have been achieved through a series of algorithmic changes for multi-threaded scenarios, data-copy optimizations, transport layer enhancements, changes in flow-control semantics and by removing bottlenecks from the code.

FioranoMQ provides the following configuration options for tuning the publish-subscribe model:

26.2.1 Setting the Message Receipt Acknowledgement

In the **DupsOkBatchSize** mode, acknowledgement of receipt of message is sent after a configurable number of messages. By default this number is 20.

The steps below enable configuration of the Message Receipt Acknowledgement parameter using the Fiorano Admin Studio in offline mode:

1. Open Fiorano Admin Studio.
2. Select **Tools > Configure Profile** from the menu bar. Select the FioranoMQ folder, and click the **Open** button. FioranoMQ is now in offline mode.
3. Navigate to **FioranoMQ -> Fiorano -> mq -> pubsub -> TopicSubSystem** in the **Profile Manager** pane. The properties of the CommonConfig are displayed in the **Properties** pane.
4. Select the **ellipsis** against the parameter **DupsOkBatchSize**. Type in the new value in the **DupsOkBatchSize** dialog box and click **OK**.
5. Right-click on the FioranoMQ domain in the **Profile Manager** pane and select the **Save** option from the shortcut menu.

26.2.2 Setting In-Memory Buffers for Subscribers

Another mechanism to manage flow-control is to increase the size of the In-Memory Buffers of topics and of subscribers with the Publisher slowdown algorithm option enabled. This will delay the initialization of publisher slowdown algorithm and increase the overall throughput. When the subscriber receives messages slowly, increasing the buffer size may lead to the Publisher slowdown algorithm not initializing. This depends on the rate at which the messages are being published on the server. This is configurable using **BasicAllowedSize** parameter through the Fiorano Admin Studio in offline mode. The default value of this variable is 128KB. The default value can be increased to a higher value depending on the scenario.

Following steps enable you to configure this parameter:

1. OpenFiorano Admin Studio.
2. Select **Tools > Configure Profile** from the menu bar. Select the FioranoMQ folder, and click the **Open** button. FioranoMQ is now in offline mode.
3. Navigate to **FioranoMQ -> Fiorano -> mq -> pubsub -> TopicSubSystem** in the **Profile Manager** pane. The properties of the PubSub Manager are displayed in the Properties pane.
4. Select the **ellipsis** against the parameter **BasicAllowedSize**. Type in the new value in the **BasicAllowedSize** dialog box and click **OK**.

5. Right-click on FioranoMQ domain in the **Profile Manager** pane and select the **Save** option from the shortcut menu.

26.2.3 Setting Parameters for New Pubsub Algorithm to Handle Slow Subscribes

In previous releases of FioranoMQ, there were no options available to disable the slowing down of the publisher when subscribers slow. In the default configuration, the publisher blocks messages by a predefined timeout and flow-control is activated.

From FioranoMQ SP2 onwards, the option of persisting messages to disk when the subscriber slows down, results in the fast subscriber not being affected even when publisher slowdown is disabled.

Below are the parameter that determines the different behaviors of the pubsub algorithm. Parameters can be configured from Fiorano Admin Studio (**FioranoMQ->fiorano->mq->pubsub->TopicSubSystem**).

Note: Pubsub parameters can be configured at either Topic level or at Subsystem level. Values taken from the topic level will be given higher priority over those taken from the Subsystem level. However, if the values at the topic level are set to the default value, then the value is fetched for the corresponding parameter from the Subsystem level itself. To set values at Topic level navigate to **FioranoMQ > Fiorano > mq > pubsub > Topics > TopicName** and change the corresponding parameters.

- **PublishBackoffThreshold:** This value indicates the size factor after which the Publisher needs to slow down. If its value is 0.6 and MaxPersistentSize is 1000MB, once the PSQ Size crosses 0.6*1000MB = 600MB then the Publisher will slow down.

  Navigate to::

  FioranoMQ->fiorano->mq->pubsub->TopicSubSystem->PublishBackoffThreshold.

  By default this value is 0.6 which means 60% of PSQ.

- **StoreMessageToDisk**: When the session buffer in the client runtime overflows, the session state in the Server becomes passive. Thepublisher starts pushing the message into temporary persistent queues in the disk (for that particular session only). The slow subscriber will not receive messages until the client's session buffer can accommodate more messages. A normal subscriber keeps receiving the message since the publisher does not slow down.

  Navigate to::

  FioranoMQ->fiorano->mq->pubsub->TopicSubSystem->StoreMessageToDisk

  The default value of this parameter is set to **yes**. The result of this parameter is not effective for non-persistent message that can be discarded.

  If the default parameter is set to **no**, this parameter ensures that messages within the FioranoMQ Server never **overflow** onto a disk queue. This means that the messages are either delivered to the subscriber (assuming the subscriber is fast enough to pick them up), or they are dropped (once the internal buffer fills up).

- **PublishWait:** indicates the default time interval by which the publisher slows down when **EnablePublisherSlowdown** is set to **yes**. As the size of the Persistent Session Queue (PSQ) changes, the publisher blocking time also changes according to the exponential back-off algorithm.

Navigate to::

FioranoMQ->fiorano->mq->pubsub->TopicSubSystem->PublishWait

The default value of this parameter is 50 milliseconds (values less than or equal to 0 are not accepted).

- **EnablePublisherSlowdown*:*** If it's set to **no**, then the publisher does not slow down when sending messages to the persistent queue (PSQ). EnablePublisherSlowdown is used when publisher flow-control is needed due to many slow subscribers. This option should be carefully selected considering the different options as mentioned above. If this option is selected the publisher will slow down if PSQ size crosses the threshold defined by PublishBackoffThreshold. as described in the description of backoff algo. The slow down time starts from the value specified by another parameter called **publishWait**. It increases according to the slow down algorithm which depends on the size of the persistent message queue for every slow subscriber. This slow down time dynamically changes as and when persistent storage size changes. This option dynamically balances publisher slow down with the size of persistent message in the disk.

  Navigate to:

  FioranoMQ->fiorano->mq->pubsub->TopicSubSystem->EnablePublisherSlowdown

  The default value of this parameter is **yes**. If the User does not want to have publisher flow-control, this parameter should be set to **no**.

- **DropOldestMessage*:*** If set to **yes**, the publisher first drops the oldest message from the persistent store and pushes the latest on the queue thereby ensuring the maximum size of persistent store is kept at all times.

  Navigated to:

  FioranoMQ->fiorano->mq->pubsub->TopicSubSystem->DropOldestMessage

  The default value of this parameter is set to **no.** If the user wants to drop the latest message, when the size of total messages in the disk cross over a certain threshold, then this parameter should be set to its default value.

- **MaxPersistentStoreSize:** Indicates the maximum size of total messages stored in the persistent queue in the disk. Once the current size crosses over this threshold, the publisher starts dropping them if DropOldestMessage flag is set to **no**. If this flag is set to **yes**, the publisher first drops the oldest message from the persistent store and pushes the latest one. If it’s value is set to **-1**, messages will be persistent till the server run out of disk space.

  Navigated to:

  FioranoMQ->fiorano->mq->pubsub->TopicSubSystem->MaxPersistentStoreSize

  The default value of this parameter is 1073741824 bytes (1 GB).

- **PublishWaitTimeChunks:** If PSQ size reaches 95% of the maximum value, the publisher publishing persistent messages will be blocked till PSQ size reduces to 85%. The publisher should wait and periodically check the PSQ sizeto confirm it has reduced. This time interval at which publisher checks the PSQ can be controlled by this parameter.

  Before checking the PSQ size, publisher will wait for MessagePublishTimeout/ PublishWaitTimeChunks in milliseconds for each iteration.

Navigated to:

FioranoMQ->fiorano->mq->pubsub->TopicSubSystem-> PublishWaitTimeChunks

The default value of this parameter is 8 (values less than or equal to 0 are not accepted).

- **IgnorePSQSizeForPersistentMessages:** This flag determines the behavior when PSQ size exceeds MaxPersistentStoreSize. If this flag is set to **true**, persistent messages will still go onto the PSQ.

  Navigated to:

  FioranoMQ->fiorano->mq->pubsub->TopicSubSystem-> IgnorePSQSizeForPersistentMessages

  The default value of this parameter is **Yes**.

- **MaxPublisherBlockTime:** When EnablePublisherSlowdown is set to yes, this indicates the maximum time for which a publisher will wait while trying to push the messages to the PSQ after it has reached 95% of its maximum limit. This applies to all publishers.

  Navigated to:

  FioranoMQ->fiorano->mq->pubsub->TopicSubSystem- >MaxPublisherBlockTime

  The default value of this parameter is 120000 milliseconds.

Points to Note:

The Pubsub algorithm needs a new profile to run the server. The older profile has to be ported to the new profile. To port the older profile to the current profile, configurations need to be replicated manually. It server is recommended that the Server run with JVM heap size with the following setting to be made in fiorano_vars.bat/.sh:

JVM_SERVER_ARGS=-server -Xms256m -Xmx512m (-Xms512m -Xmx1024m for enterprise operation)

Non-persistent messages will always be dropped if PSQ size exceeds MaxPersistentStoreSize. However, non-persistent messages can persist till the Server goes out of disk space by setting its value to -1.

Flow control: Backoff algorithm will kick-in if PSQ size crosses PublishBackoffThreshold. By default, when 60% of the PSQ is used, every published call will be blocked for PublishWait (50ms by default). With every 5% increase in the PSQ size, publish call blocking time will be doubled. At the same time, the publish call blocking time will be halved with every 5% decrease. If 95% of the PSQ is used, the publish call will block calls for an infinite time. The publish call blocking time will reduce to 1600 ms if the PSQ size goes down to 85%.

The new back off algorithm can be disabled by setting EnablePublisherSlowdown to **No**.

The table below gives the default Blocking Time during various states of PSQ:

% of PSQ Used	**Publish Block Time**
60 %	50 ms (PublishWait default)
65 %	100 ms
70 %	200 ms
75 %	400 ms
80 %	800 ms
85 %	1600 ms
90 %	3200 ms
95 %	Block till PSQ size reduces to 85%

26.3 Calculating Memory Requirements for FioranoMQ Server

Approximate minimum memory allocated by default for each module in the FioranoMQ Server is given in the table below:

Module	**Minimum Amount of Memory Required**
Topic Connection	128 KB This is the default connection buffer size associated with each topic connection. This value is determined by the parameter BasicAllowedSize in the Topic Subsystem. More information about this parameter can be found in **Chapter 4: Topic Subsystem Level Configuration** [Link?] of the *FioranoMQ Reference Guide*. Please see Section 26.2.2 Setting in-memory buffers for Subscribers for how to change this parameter.
Topic Manager (Unique for all topics)	512 KB This is the default maximum in-memory size of the Message buffer that is used across all the storage topics in FioranoMQ. This value is determined by the parameter MaxBufferSize in the GMS Topic Manager. This can be set using Admin Studio in Offline editing mode by navigating to Fiorano > mq > pubsub > databases > file > TgmsManager > MaxBufferSize.
Queue	1024 KB (Non-persistent buffer size) This is the default buffer size for the Non-Persistent Buffer associated with each storage queue. This value is determined by the parameter InMemoryBufferSize in the Queue Subsystem/Queue. More information about this parameter can be found in **Chapter 3: Queue Subsystem Level Configurations** [Link?] of the *FioranoMQ Reference Guide*. Please see section 29.1 Support for Destination Level Configuration for referencing the difference between setting a parameter value at Subsystem level and at the Destination level. 512 KB (Persistent buffer size)

Module	**Minimum Amount of Memory Required**
	This is the default buffer size for the Persistent Buffer associated with each storage queue. This value is determined by the parameter PersistentInMemoryBufferSize in the Queue Subsystem/Queue. More information about this parameter can be found in **Chapter 3: Queue Subsystem Level Configurations** of the *FioranoMQ Reference Guide*. Please see section 29.1 Support for Destination Level Configuration for referencing the difference between setting a parameter value at Subsystem level and at the Destination level.
Queue Receiver	256 KB This is the default maximum buffer size associated with each Queue Receiver when retrieving the messages from the Server DB. This value is determined by the parameter MaxPrefetchSize in the Queue Subsystem. More information about this parameter can be found in **Chapter 3: Queue Subsystem Level Configurations** of the *FioranoMQ Reference Guide*. Please see section 26.1.1.5 PTP Prefetch size to learn how to change this parameter and to learn of its importance.
Route	128 KB This is the default buffer size associated with the Route subscriber. This parameter comes into use when a Route is created from a Topic (Source) to a Queue (Target). This value is determined by the parameter MaxTopicBuffer of the Route Manager. More information about this parameter can be found in **Chapter 15: Advance Configuration** [Link?] of the *FioranoMQ Reference Guide*.
Total	2560 KB

The minimum amount of memory consumed by the JMS objects inside FioranoMQ Server JVM can be calculated using the formula below:

Minimum memory required = (Number of topic connections * 128) + 512 + (Number of Queues * 1536) + (Number of Queue Receivers * 256) + (Number of Routes * 128) (In Kilobytes).

Note: These calculations are only done to determine the minimum amount of memory consumed by the JMS Objects created inside the FioranoMQ Server. This is not sufficient amount of memory to deploy the FioranoMQ Server. The amount of memory necessary to deploy the FioranoMQ Server is generally higher than the value obtained in these calculations.

Chapter 27: Administrating the FioranoMQ Server Using APIs

27.1 Introduction

FioranoMQ provides a comprehensive set of Administration APIs that allow the Enterprise Administrator to access MQ services such as MQNamingService, MQAdminService, MQRealmService, MQSnooperService and MQDispatcherService.

Most of the functions provided by Admin APIs can be accessed using the Admin GUI.

27.2 Creating an Admin Connection

The first step in using AdminService involves obtaining a handle to an Admin connection. The handle to an Admin connection is obtained as by entering:

```
Hashtable env = new Hashtable();

env.put(Context.SECURITY_PRINCIPAL, "admin");

env.put(Context.SECURITY_CREDENTIALS, "passwd");

env.put(Context.PROVIDER_URL, "http://localhost:1856");

env.put(Context.INITIAL_CONTEXT_FACTORY,
"fiorano.jms.runtime.naming.FioranoInitialContextFactory");

InitialContext ic = new InitialContext(env);

System.out.println("Created InitialContext :: " + ic);

MQAdminConnectionFactory acf = (MQAdminConnectionFactory) ic.lookup("primaryACF");

System.out.println("Looked up MQAdminConnectionMetaData :: " + acf.getMeta-

Data());

System.out.println("Looked up MQAdminConnectionFactory :: " + acf);

MQAdminConnection ac = acf.createMQAdminConnection("admin", "passwd");

System.out.println("Created Admin Connection :: " + ac);
```

The default Admin password is set to “passwd”. The FioranoMQ 9 Administrator can change “passwd” using the changePassword API. MQAdminConnection can be used to obtain handles to the FioranoMQ 9 Administrative services discussed in the preceding sections.

Once the admin connection has been made to the MQServer, the administrator has access to services explained in the following sub-sections:

27.2.1 MQNamingService

FioranoMQ Naming Services provide a JNDI implementation to lookup all the Administered Objects. By default, the Administered Objects are stored in FioranoMQ's persistent store.

```
MQNamingService namingService = ac.getMQNamingService();
```

Upon execution of this step, the User can invoke all APIs avaliable in MQNamingService.

For example:

```
lookup(String adminObjectName);

bind(String adminObjectName, Object data);
```

Note: For details of available, APIs, please refer to API documentation available at %FIORANO_HOME\fmq\docs\api\index.html

27.2.2 MQAdminService

MQAdminService allows the MQ Administrator to create/delete/edit administered objects such as Destinations and ConnectionFactories.

```
MQAdminService adminService = ac.getMQAdminService();
```

Upon execution of this step, the User can invoke all APIs avaliable in MQAdminService.

For example:

```
// CREATION OF TOPICS

System.out.println("\n ** Creating Topics ** \n");

try

{

// TOPIC #1

//Create the TopicMetaData with the desired Topics

TopicMetaData metadata1 = new TopicMetaData();

metadata1.setName("Mytopictest1");

adminService.createTopic(metadata1);

System.out.println("Successfully created Topic:: " + metadata1.getName());
}

catch (JMSException e)

{

System.out.println(e);

}
```

Note: For details of available APIs, please refer to API documentation available at %FIORANO_HOME\fmq\docs\api\index.html.

27.2.3 MQRealmService

RealmServices provide complete integration with existing NT/Solaris realms. Topic ACL/ACE settings for Users can be set and viewed using MQRealmService APIs. In addition to ACL creation, a Realm service can be used for UserManagerment.

The sample below demonstrates the logic for the creation of new Users in FioranoMQ:

```
MQRealmService realmservice = ac.getMQRealmService();
```

Upon execution of this step, the User can invoke all APIs avaliable in MQRealmService.

For example:

```
// CREATION OF USERS

System.out.println("\n ** Creating Users ** \n");

try

{

Principal user1 = realmservice.createUser("Bill", "Clinton");

Principal user2 = realmservice.createUser("Rod", "Steward");

System.out.println("Created Users");

}

catch (JMSException e)

{

System.out.println(e);

}
```

Note: For details of available, APIs please refer to API documentation available at %FIORANO_HOME\fmq\docs\api\index.html.

27.2.4 MQSnooperService

MQ Administrator can install snoopers on selected topics/Queues. By using MQSnooperService APIs, details of all the published messages can be viewed. Destinations can be registered and unregistered for snooping at runtime using the APIs in MQSnooperService.

MQSnooperService snooperService = ac.getMQSnooperService();

Upon execution of this step, the User can invoke all APIs avaliable in MQSnooperService.

For example:

The following API checks if the given JMS destination has been registered for Snooping.

boolean isDestinationRegistered(String destinationName);

Note: For details of available, APIs please refer to API documentation available at %FIORANO_HOME\fmq\docs\api\index.html.

27.2.5 MQDispatcherService

MQDispatcherService APIs allow the MQ Administrator to set up the Server as a dispatcher, route requests to FioranoMQ Servers in a cluster and view dispatcher status. The DispatcherService allows the administrator to add/remove Servers to the Cluster, locating the Preferred Server in the Cluster at runtime.

MQDispatcherService dispatcherService = ac.getMQDispatcherService();

Upon execution of this step, the User can invoke all APIs avaliable in MQDispatcherService.

For example:

public ServerMetaData getPreferredServer()

Note: For details of available, APIs please refer to API documentation available at %FIORANO_HOME\fmq\docsapiindex.html.

27.2.6 MQMonitoringService

MQMonitoringService APIs allow the FioranoMQ administrator to control the events that are used to monitor the status of the FioranoMQ Server. It is possible to start and stop monitoring events using this service. Additionally, monitoring for specific events can be added and removed.

MQMonitoringService dispatcherService = ac.getMQMonitoringService();

Upon execution of this step, the User can invoke all APIs avaliable in MQMonitoringService.

For example:

public void startSystemEvents()

Note: For details of available, APIs please refer to API documentation available at %FIORANO_HOME\fmq\docs\api\index.html.

Note: Samples illustrating the use of Admin APIs can be found in the %FIORANO_HOME%\fmq\samples folder of the FioranoMQ installation directory.

* MQTraceService and MQLogService are not supported by FioranoMQ upwards.

Chapter 28: DB Recovery Tool

The Fiorano DB Recovery Tool is used to recover the corrupted database of FioranoMQ Server's file.

Warning: If the Database is changed manually it gets corrupted and the DB Recovery Tool will not be able to recover any data.

28.1 Overview of FioranoMQ's file based Database

The database section consists of three main sections:

1. **PTP:** This contains all the data related to the PTP subsystem.
2. **PubSub:** This contains all the data tables and subscriber information tables related to the PubSub subsystem. PubSub also contains topic-wise directories for Persistent Messages as well as Non Persistent Message, for both data and subscriber information.
3. **SDB:** This contains all the data related to the Security subsystem which consists of all Users, their permissions, Acls and Groups they belong to.

28.2 Typical Structure of FioranoMQ File Based DB

```
FioranoMQ
    Run
        PTP
            CeMaster.tbl
            QGMS.PRIMARYQUEUE.4
                CeTable.tbl
                Ce1.dat
                Ce2.dat
                ....
            QGMS.PRIMARYQUEUE.5
                ....
        PUBSUB
            CeMaster.tbl
            JMSX_PM_TIDPRIMARYTOPIC.data
```

```
                CeTable.tbl

                Ce1.dat

                Ce2.dat

                ....

        JMSX_PM_TIDPRIMARYTOPIC. subscribersInfo

                CeTable.tbl

                Ce1.dat

                Ce2.dat

                ....

        JMSX_NPM_TIDPRIMARYTOPIC.data

                ....

        JMSX_NPM_TIDPRIMARYTOPIC. SubscribersInfo

                ....

        JMSX_PM_TIDSECONDARYTOPIC.data

                ....

SDB

        REALM.ACL

                CeMaster.tbl

                ACL_TABLE

                        CeTable.tbl

                        Ce1.dat

                        Ce2.dat

                        ....

                PRINCIPAL_TBL

                        CeTable.tbl

                        Ce1.dat

                        Ce2.dat
```

```
                    ....
            REALM.PRINCIPAL
                    CeMaster.tbl
                    MEMBER_TBL
                            CeTable.tbl
                            Ce1.dat
                            Ce2.dat
                            ....
                    PRINCIPAL_TBL
                            CeTable.tbl
                            Ce1.dat
                            ....
```

28.3 Using FioranoMQ DB Recovery Tool

The DB Recovery Tool is executed by running recover-database.bat/ recover-database.sh which can be found under *%FIORANO_HOME%\fmq\bin*.

28.3.1 Using Windows

```
> recover-database.bat [–propertiesFile <absolute path for configuration file>]
              [-fmq.profile <profile name for which DB is to be recovered>]
              [-h ]
```

28.3.2 Using Unix/Linux

```
$ ./recover-database.sh [–propertiesFile <absolute path for configuration file>]
              [-profile <profile name for which DB is to be recovered>]
              [-h ]
```

28.3.3 Parameters

1. **Properties file:** Configuration file. Default configuration file is located at %FIORANO_HOME%\fmq\profiles\recovery.properties

Example:

```
> recover-database.bat –propertiesFile  c:\db.properties
```

2. **Profile:** Profile name for which DB is corrupted and to be recovered.

   Example:

   ```
   > recover-database.bat  –profile FioranoMQ_XA
   ```

3. **Help:** Brief information onvarious command line arguments.

   Example:

   ```
   > recover-database.bat  -h
   ```

4. **DB Path**: Path in which FMQ DB is located.

   Example:

   ```
   > recover-database.bat -dbPath %FIORANO_HOME%\fmq\profiles\FioranoMQ_XA\run
   ```

5. **checkCSP**: Analyzes CSP cache specified in path

   Example:

   ```
   > recover-database.bat -checkCSP  %CSP_CACHE_PATH%
   ```

6. **Operation**: The operation to be performed by the recovery tool. By default the value is RunExtractor.

   Example:

   ```
   > recover-database.bat RunExtractor
   ```

28.3.4 Configuration File Parameters

1. **RUN_PTP_ANALYZER:** Analyzes whether or not to recover the PTP subsystem database. The default is **true**.
2. **RUN_PUBSUB_ANALYZER:** Analyzes whether or not to recover PUBSUB subsystem database. The default is **false**
3. **RUN_SDB_ANALYZER:** Analyzes whether or not to recover Security subsystem database. The default is **false**.
4. **TRACE_LEVEL:** Trace level for DB recovery tool. The default number is 6 (which is the maximum).
5. **PTP_DB_NAME:** This is the database name for the PTP Subsystem. The default value is PTP. This default value can be configured from **Profile Manager** by setting the value of **Fiorano.mq.ptp.databases.file.FileDBManager.Path**
6. **PUBSUB_DB_NAME:** This is the database name for the PUBSUB SubSystem. The default value is PUBSUB. This default value can be configured from **Profile Manager** by setting the value of **Fiorano.mq.pubsub.databases.file.FileDBManager.Path**

7. **SDB_ACL_DB_NAME:** This is the database name for SDB ACL Manager. The default value is SDB/REALM.ACL. This default value can be configured from **Profile Manager** by setting the value of **Fiorano.security.AclManager.FileDBManager.Path**
8. **SDB_PRINCIPAL_DB_NAME:** This is the database name for SDB Principal Manager. The default value is SDB/REALM.PRINCIPAL. This default value can be configured from **Profile Manager** by setting the value of **Fiorano.security.PrincipalManager.FileDBManager.Path**
9. **RENAME_DIRECTORY ANALYZER:** Analyzes whether or not to rename the recovered directory to the default name. The default is **false**.

   Default names for PTP, PUBSUB and Security subsystem database are PTP, PUBSUB and SDB respectively.

   When 'tool' recovers some of the database it creates a new directory by adding prefix .**New** to the default name, that is, **PTP.New**.

   If RENAME_DIRECTORY is **true**, it renames the old (may be corrupted) database with prefix **.Old**, that is, **PTP.Old,** and renames the recovered database to the default state by removing prefix **.New**.

28.4 Steps to Run DBRecovery Tool

28.4.1 Parameter Configuration and Execution

1. Set the required configuration file parameters in the default configuration file $FIORANO_HOME/fmq/profiles/recovery.properties or in the user defined file (absolute path of the configuration file should be explicitly mentioned as parameter –propertiesFile <path of the file> while running the tool).
2. Execute the DB Recovery tool by running the recover-database.bat/ recover-database.sh which can be found under $FIORANO_HOME/fmq/bin with the necessary parameters.

   Example: Using Unix/Linux:

   $ ./recover-database.sh RunExtractor -profile FioranoMQ_XA -dbPath $FIORANO_HOME/fmq/profiles/Fiorano_XA/run
3. Now the user can verify the results of the DB analysis through the log files which will get generated in $FIORANO_HOME/fmq/bin.

   Example:

   Consider the analysis has been done for PTP DB.

   $FIORANO_HOME/fmq/bin/ptpResults.log

   $FIORANO_HOME/fmq/bin/ptpUnableToRecoverData.log

28.4.2 DBRecovery

1. On setting the RECOVER_DB property to true i.e. **RECOVER_DB=true** (in fmq/profiles/recovery.properties file) will not only analyze the DB but also recover the DB if found corrupted.

2. The recovered DB entry/table files will be generated in a new directory which will be created by adding suffix .New to the default name, that is, PTP.New, if the value of RENAME_DIRECTORY property is set to false (in fmq/profiles/recovery.properties file). Then the user has to manually rename the original DB directory with suffix .Old i.e.PTP to PTP.Old and the newly recovered DB directory by removing the suffix .New to default i.e. PTP.New to PTP.
3. If the RENAME_DIRECTORY property is set to true then by default, the corrupted DB will be renamed with suffix .Old i.e. PTP to PTP.Old and also renames the recovered database to default by removing the suffix .New i.e. PTP.New to PTP .No manual renaming is required.
4. If the user set the RECOVER_DB property to false i.e. **RECOVER_DB=false** (in fmq/profiles/recovery.properties file) then only the analysis of DB will be performed and the results will be logged.

Chapter 29: Application Server Integration

29.1 Implementing Advanced JMS APIs

In this JMS-Application Server integration the JMS provider implements advanced JMS APIs. These APIs deliver messages to consumers within the context of an Application Server session.

Download the sample (AppServer-JMS Sample Applications) that illustrates the utility of JMS APIs, which allow Application Server Sessions to be registered with the FioranoMQ JMS Server. Messages are delivered to the consumers within the context of the Application Server Session.

29.2 Message Driven Beans

FioranoMQ integrates with most Application Servers to provide the **Message Driven Bean** function.

Message-Driven Beans (MDB's) are stateless, server-sided and transaction-aware components used to process asynchronous JMS messages. Introduced in EJB 2.0, message-driven beans process messages that are delivered through the Java Message Service. MDBs can receive JMS messages and process them. While a MDB is responsible for processing messages, its container automatically manages the entire environment of the component including transactions, security, resources, concurrency and message acknowledgment.

29.3 FioranoMQ - EJB Application Server Integration

FioranoMQ provides an Application Server neutral mechanism to integrate JMS with EJB. This document reviews in depth the issues that FioranoMQ has solved with this method of integration. Samples illustrating how JMS can be used for asynchronous invocation of EJB methods can be downloaded from JMS-EJB Sample Applications. This section explains how the power of EJB can be leveraged using asynchronous Java Messaging technology. This section also illustrates how FioranoMQ's implementation of JMS provides support for asynchronous EJB method invocation, as well as the limitations inherent in the EJB architecture for doing the same. It also contains an EJB JMS sample.

Note: The reader must be familiar with the basic concepts and workings of JMS and EJB.

29.3.1 Asynchronous Method Invocation using Delegation Model

Using the services provided by FioranoMQ, it is possible for one EJB to asynchronously invoke methods on another Bean. FioranoMQ achieves this through a delegation model, where a request is passed on, or delegated to, another entity that handles the request and returns a response. This allows a given EJB (or a Client Application using EJB) to trigger methods on any other Bean. The delegator class is responsible for listening on a JMS Queue, marshalling the incoming JMS messages and invoking appropriate remote methods on the desired target Bean. The delegator class can be executed as either an Application Server specific startup class or as a standalone JMS Application. If the delegator class is assigned as one of the startup classes of the Application Server, then the Application Server is responsible for invoking this class in the JVM instance.

Note: It is not necessary for the delegator class to be present in the startup group of classes of the Application Server. The startup classes are Application Server specific making the code non-portable.

Therefore, the FioranoMQ EJB JMS sample that illustrates the EJB integration has not made the delegator class a member of the startup class of any application server. This sample illustrates how FioranoMQ can be integrated with any Application Server through a single standalone application. Please refer to the documentation related to the 'Application Server' to determine how the delegator class can be made a startup class.

29.3.2 EJB JMS Sample Application

The example in the figure below illustrates how JMS can be used to invoke, asynchronously, methods on an EJB.

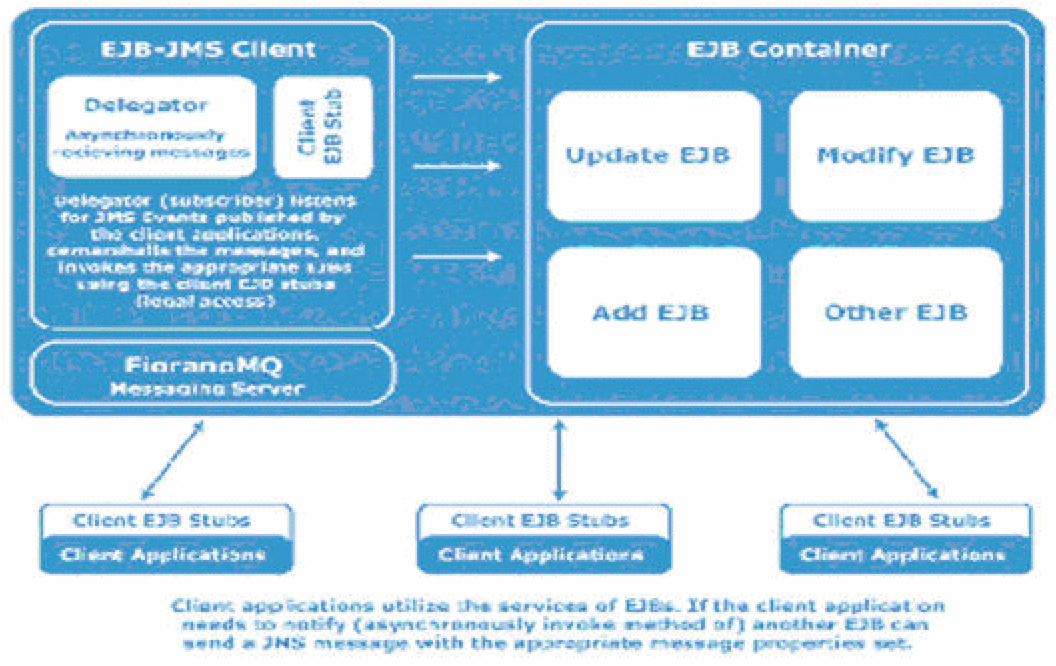

EJB-JMS Client
Delegator
Asynchronously receiving messages
Client EJB Stub
Delegator (subscriber) listens for JMS Events published by the client applications, demarshalls the messages, and invokes the appropriate EJBs using the client EJB stubs (local access)
FioranoMQ Messaging Server
EJB Container
Update EJB
Modify EJB
Add EJB
Other EJB
Client EJB Stubs
Client Applications
Client EJB Stubs
Client Applications
Client EJB Stubs
Client Applications
Client applications utilize the services of EJBs. If the client application needs to notify (asynchronously invoke method of) another EJB can send a JMS message with the appropriate message properties set.

Figure1: Illustrating the Use of JMS for Asynchronous Method Invocation on EJBs

The sample and the JMS-EJB integration work as listed below:

- Client applications (referred to as **Client EJB Stubs**) publish a JMS message each time they want to invoke a method on a remote EJB
- All published JMS messages are received by a delegator application, which invokes the appropriate methods on the remote EJB using the Java reflection API.
- The JMS-EJB integration application has three components, each of which is discussed below:
    - **Event Generator**: The Client application generating the requests to invoke the remote EJB methods
    - **Delegator**: A FioranoMQ JMS application that reads the message and invokes each remote EJB method
    - **EJBs**: The actual EJBs whose methods are invoked

29.3.3 Event Generator

The Event Generator publishes JMS Messages to a Queue (**primaryqueue** by default). Each message contains relevant information regarding the methods to be invoked in specific, remote, EJBs. The Client Application represents the Event Generator. In a real world example, the Event Generator might be a **Shopping cart** Bean that needs to asynchronously trigger the method of another Bean, dependent upon the User clicking **checkout** on an HTML page.

The file Publisher.java contains the Event Generator, which simulates an event that is used to invoke appropriate EJBs. The goal of the Generated Event is to invoke one of the three beans (Add, Delete or Update), once every two seconds. Since there are no limitations to the JMS Messages being published by EJB, the Event Generator, in effect, becomes an EJB. The only task of the Event Generator is to generate JMS messages that encapsulate the Add/Delete/Update events which are then invoked on a remote bean.

The JMS message (event) includes the following information:

- Name of the target EJB to be looked up.
- The API used to create instances of the EJB and arguments, if any.
- Name of the method to invoke in the target EJB and arguments, if any.
- The information above is required by the Delegator class to invoke the appropriate method asynchronously on the target bean.

29.3.4 Delegator

The Delegator represents a generalized application that is responsible for receiving events asynchronously and invoking appropriate methods on the target EJB. The JMS Application listens for messages on a queue (**primaryqueue** by default), de-marshals the messages and invokes the appropriate method in the target EJB. The Delegator class is a generalized class that uses reflection to invoke any remotely accessible method in the requested EJB. The class Subscriber.java represents the Delegator class. The JMS Application implements an asynchronous listener to listen in for messages that are published on the queue **primaryQueue**. The Delegator receives the JMS Messages that are published on the Queue and demarshalls the content of the JMS Message. It finally looks up and invokes appropriate methods on the EJB with appropriate parameters, using reflection. Optionally, this class can be placed in the startup class of an Application Server, thereby passing on the responsibilities of the Delegator to the Application Server.

29.3.5 Enterprise Java Beans

Three simple stateless session Beans (AddBean, UpdateBean and SessionBean) are used as the target EJBs. The method of the Beans is invoked by the Delegator (described in section 29.3.4 Delegator).

29.3.6 Limitations of Enterprise Java Beans

EJBs do not provide for asynchronous method invocation for the reasons given below:

1. EJBs do not run as a daemon service. EJBs are server-side reusable components that can be put back into the available pool of the container when they time-out or when there are no more active references to the Bean.
2. EJBs can be accessed only through a remote interface. An EJB cannot be made to receive a JMS Message asynchronously, since this result in the direct invocation of Bean methods. By definition, an EJB can only be accessed through its remote interface. The container of the EJB is responsible for managing and invoking all calls on the Bean. Since the EJB container manages all transactions and threads the safety operations of the Bean, direct access to the Bean method is restricted, as this can result in a potentially catastrophic operation. EJBs cannot be used to create a JMS message listener that asynchronously receives messages since this would invocation the Bean methods directly.

29.4 FioranoMQ Client Logging

For FioranoMQClientRTL logging to take place, the logger levels and appenders have to be mentioned through properties file or xml file. The jboss application server uses log4j.xml file. In this case, the FioranoMQClientRTL loggers' level and appender has to be mentioned in the same log4j.xml.

Example: If the logger name is "Fiorano.FMQ.Services.ClientRootLoggerServices.FMQClientLoggerServices", the logger level is INFO and the log appender is RollingFileAppender, then

```
<category
name="Fiorano.FMQ.Services.ClientRootLoggerServices.FMQClientLoggerServices">

    <priority value="INFO"/>

    <appender-ref ref="FioranoFileAppender"/>

</category>



<appender name="FioranoFileAppender" class="org.apache.log4j.RollingFileAppender">

    <errorHandler class="org.jboss.logging.util.OnlyOnceErrorHandler"/>

    <param name="File" value="${jboss.server.log.dir}/log.out"/>

    <param name="MaxFileSize" value="10MB"/>

    <param name="MaxBackupIndex" value="5"/>

    <param name="Append" value="true"/>

        <layout class="org.apache.log4j.PatternLayout">

      <param name="ConversionPattern" value="%d %-5p [%c] %m%n"/>

        </layout>

</appender>
```

log4j.properties file will contain the logger names used in FioranoMQClientRTL. By default, "log4j.properties" file will be provided under %FIORANO_HOME%\fmq\bin.

Note the following:

1. For information regarding Appender,MaxFileSize, MaxBackupIndex,Append,Levels and layout , please refer **Chapter 8: Logger Configuration** in *FioranoMQ Reference Guide*.
2. For information regarding log4j.properties configurations and Parent-child relationship in FioranoMQ Client Logger, please refer Chapter 21: Logger.

29.5 Integrating FioranoMQ with J2EE Servers

FioranoMQ provides a standardized method to connect and integrate with a J2EE container using the FioranoMQ Resource Adapter. This Adapter enhances the plug-in ability and integration of FioranoMQ with J2EE Servers.

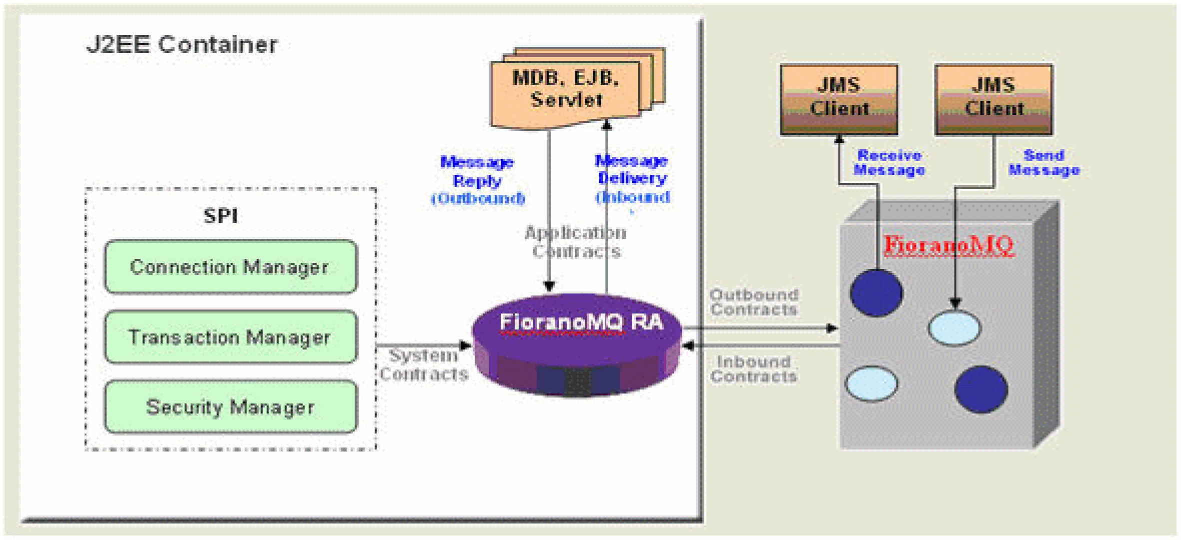
The FioranoMQ Resource Adapter implements the complete semantics of JCA 1.5 allowing:

3. J2EE applications (MDBs and EJBs) to send messages to JMS Topics and Queues using a unified JMS model with JCA Outbound Contracts (including Transaction Contracts).

4. Message endpoints (MDBs) to asynchronously receive messages from Topics/Queues using JCA 1.5 Message Inflow Contracts.
5. The resource adapter to propagate Transactions initiated by the FioranoMQ Server to the application server, using Transaction Inflow contracts.

29.5.1 How Resource Adapter Works

The working of the Resource adapter is illustrated in the figure below:



29.5.2 Deployment of FioranoMQ Resource Adapter

The FioranoMQ Resource Adapter has been packaged in a **rar** file that can be used to deploy the Resource Adapter:

6. %FMQ_HOME%\fmq\lib\jca\fmq-connector-ra.rar can be used to deploy the FioranoMQ Resource Adapter in J2EE 1.3 compliant Application Servers.
7. %FMQ_HOME%\fmq\lib\jca\fmq-connector-ra.rar can be used to deploy The FioranoMQ Resource Adapter in J2EE 1.4 compliant Application Servers.

Note: This is where %FMQ_HOME% represents the installation directory of FioranoMQ.

In the FioranoMQ Resource Adapter's deployment descriptor (ra.xml), the properties, below, can be configured/overwritten during deployment:

S.No	Type	Description	Property
1	ProviderURL	String	Provider URL (where JMS admin objects are stored).
2	BackupProviderURLs	String	Backup provider URLs, if any.

S.No	Type	Description	Property
3	Initial Context-Factory	String	Name of the class providing the implementation of Initial Context Factory.
4	JndiUserName	String	Username to be used for lookup.
5	JndiPassword	String	Password to be used for lookup.
6	ConnectionFactory	String	Name of the connection factory to be used for creating a physical connection with the FioranoMQ Server.
7	ClientID	String	ID to identify the JMS client (message consumer).
8	JmsUserName	String	Username to be used for creating a connection to the FioranoMQ Server.
9	JmsPassword	String	Password to be used for creating a connection to the FioranoMQ Server.
10	XAEnabled	Boolean	Whether or not to enable XA for receiving messages.
11	DebugEnabled	Boolean	Whether or not to print debug statements. To log debug statements RA tries to create an instance of java.util.logging.Logger. If unable to create Logger then RA writes all debug statements using System.out

The above properties are used by the FioranoMQ Resource adapter for implementing Message-Inflow support. Some of these properties can be overridden by Activation Configuration, as described in the section below.

29.5.3 Configuring the Resource Adapter

<Inbound-resourceadapter> configuration

Inbound Communication allows the Resource Adapter (FioranoMQ RA) to handle messages flowing in from the EIS (JMS Server) to the application (MDB) residing in an application server. The FioranoMQ Resource Adapter (RA) uses the Generic Message Inflow Contract and asynchronously delivers messages to message driven beans via the onMessage () method using the javax.jms.MessageListener interface.

The MDB, when deployed, is registered as a message endpoint for receiving the messages from the Resource Adapter (RA). On deployment of the MDB, the Deployer provides Activation-Configuration information to the RA so that it can deliver messages to the MDB.

Note: For complete details on semantics with which the Message-Driven Bean asynchronously receives messages using FioranoMQ RA, please refer to *Section 12.7.1* of *JCA 1.5 specifications.*

FioranoMQ RA specifies the following Activation-Configurations in the FioranoMQ-RA deployment descriptor:

More details on the Activation-Configuration for Message Inflow to JMS endpoints can be found in *Appendix B* of *JCA 1.5* specification document.

FioranoMQ RA uses JMS Application Server Facilities for concurrent processing of subscription messages and message delivery to the MDB endpoints.

Note: For details on JMS Application Server Facilities, refer to *Chapter 8 of JMS 1.1* specifications.

29.5.3.1 Sample Activation Configuration

Sample Configurations to be specified in ra.xml

```
<config-property>

      <config-property-name>ProviderURL</config-property-name>

      <config-property-type>java.lang.String</config-property-type>

      <config-property-value>http://localhost:1856</config-property-value>

     </config-property>

     <config-property>

      <config-property-name>BackupProviderURLs</config-property-name>

      <config-property-type>java.lang.String</config-property-type>

      <config-property-value></config-property-value>

     </config-property>

     <config-property>

      <config-property-name>InitialContextFactory</config-property-name>

      <config-property-type>java.lang.String</config-property-type>

      <config-property-
value>fiorano.jms.runtime.naming.FioranoInitialContextFactory</config-property-value>
```

```
</config-property>

<config-property>

  <config-property-name>JndiUserName</config-property-name>

  <config-property-type>java.lang.String</config-property-type>

  <config-property-value>anonymous</config-property-value>

</config-property>

<config-property>

  <config-property-name>JndiPassword</config-property-name>

  <config-property-type>java.lang.String</config-property-type>

  <config-property-value>anonymous</config-property-value>

</config-property>

<config-property>

  <config-property-name>ConnectionFactory</config-property-name>

  <config-property-type>java.lang.String</config-property-type>

  <config-property-value>primaryTCF</config-property-value>

</config-property>

<config-property>

  <config-property-name>ClientID</config-property-name>

  <config-property-type>java.lang.String</config-property-type>

  <config-property-value></config-property-value>

</config-property>

<config-property>

  <config-property-name>JmsUserName</config-property-name>

  <config-property-type>java.lang.String</config-property-type>

  <config-property-value>anonymous</config-property-value>

</config-property>

<config-property>
```

```
      <config-property-name>JmsPassword</config-property-name>
      <config-property-type>java.lang.String</config-property-type>
      <config-property-value>anonymous</config-property-value>
    </config-property>
    <config-property>
      <config-property-name>XAEnabled</config-property-name>
      <config-property-type>java.lang.Boolean</config-property-type>
      <config-property-value>false</config-property-value>
    </config-property>
    <config-property>
      <config-property-name>DebugEnabled</config-property-name>
      <config-property-type>java.lang.Boolean</config-property-type>
      <config-property-value>true</config-property-value>
    </config-property>
```

Definition for inbound and outbound resource adapters are given in ra.xml

<inbound-resourceAdapter> in ra.xml

```
<inbound-resourceadapter>
      <messageadapter>
        <messagelistener>
          <messagelistener-type>
            javax.jms.MessageListener
          </messagelistener-type>
          <activationspec>
            <activationspec-class>
              com.fiorano.mq.ra.FMQActivationSpec
            </activationspec-class>
            <required-config-property>
```

```
  <config-property-name>destinationName</config-property-name>
</required-config-property>
<required-config-property>
  <config-property-name>destinationType</config-property-name>
</required-config-property>
<required-config-property>
  <config-property-name>messageSelector</config-property-name>
</required-config-property>
<required-config-property>
  <config-property-name>acknowledgeMode</config-property-name>
</required-config-property>
<required-config-property>
  <config-property-name>subscriptionDurability</config-property-name>
</required-config-property>
<required-config-property>
  <config-property-name>subscriptionName</config-property-name>
</required-config-property>
<required-config-property>
  <config-property-name>clientID</config-property-name>
</required-config-property>
<required-config-property>
  <config-property-name>maxSessions</config-property-name>
</required-config-property>
<required-config-property>
  <config-property-name>userName</config-property-name>
</required-config-property>
<required-config-property>
```

```
                <config-property-name>password</config-property-name>

              </required-config-property>

            </activationspec>

          </messagelistener>

        </messageadapter>

      </inbound-resourceadapter>
```

Configurations for inbound flow are specified in deployment descriptor 'ejb-jar.xml' located in MDB's META-INF folder in case of EJB2 and specified using annotations in case of EJB3.

Sample Activation configuration using EJB2.

<ejb-jar.xml> present in doc.

```
<ejb-jar xmlns="http://java.sun.com/xml/ns/j2ee"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee

http://java.sun.com/xml/ns/j2ee/ejb-jar_2_1.xsd"

version="2.1">

<enterprise-beans>

<message-driven>

<ejb-name>FMQSampleMDB</ejb-name>

<ejb-class>com.fiorano.mq.ra.mdb.FMQSenderMDB_1_4</ejb-class>

<messaging-type>javax.jms.MessageListener</messaging-type>

<transaction-type>Container</transaction-type>

<activation-config>

        <activation-config-property>

                <activation-config-property-name>destinationName</activation-config-
propertyname>

                <activation-config-property-value>primaryTopic</activation-config-
propertyvalue>

        </activation-config-property>

        <activation-config-property>
```

```
                <activation-config-property-name>destinationType</activation-config-
propertyname>

                <activation-config-property-value>javax.jms.Topic</activation-config-
propertyvalue>

        </activation-config-property>

        <activation-config-property>

                <activation-config-property-name>messageSelector</activation-config-
propertyname>

                <activation-config-property-value></activation-config-property-value>

        </activation-config-property>

        <activation-config-property>

                <activation-config-property-name>acknowledgeMode</activation-config-
propertyname>

                <activation-config-property-value>Auto-acknowledge</activation-config-
propertyvalue>

        </activation-config-property>

        <activation-config-property>

                <activation-config-property-name>subscriptionDurability</activation-
configproperty-name>

                <activation-config-property-value>false</activation-config-property-value>

        </activation-config-property>

        <activation-config-property>

                <activation-config-property-name>subscriptionName</activation-config-
propertyname>

                <activation-config-property-value>test</activation-config-property-value>

        </activation-config-property>

        <activation-config-property>

                <activation-config-property-name>clientID</activation-config-property-
name>

                <activation-config-property-value>test</activation-config-property-value>

        </activation-config-property>
        <activation-config-property>

                <activation-config-property-name>maxSessions</activation-config-
propertyname>

                <activation-config-property-value>0</activation-config-property-value>

        </activation-config-property>

        <activation-config-property>

                <activation-config-property-name>userName</activation-config-property-
name>

                <activation-config-property-value>anonymous</activation-config-property-
value>

        </activation-config-property>

        <activation-config-property>

                <activation-config-property-name>password</activation-config-property-
name>

                <activation-config-property-value>anonymous</activation-config-property-
value>

        </activation-config-property>

</activation-config>

</message-driven>

</enterprise-beans>

</ejb-jar>
```

Sample Activation configuration using EJB3 annotations

```
@MessageDriven(name = "FMQSampleMDB", activationConfig = {
        @ActivationConfigProperty(propertyName = "destinationName", propertyValue =
"primaryTopic"),
        @ActivationConfigProperty(propertyName = "destinationType", propertyValue =
"javax.jms.Topic"),
        @ActivationConfigProperty(propertyName = "messageSelector", propertyValue = ""),
        @ActivationConfigProperty(propertyName = "subscriptionDurability", propertyValue =
"false"),
        @ActivationConfigProperty(propertyName = "subscriptionName", propertyValue =
"test"),
        @ActivationConfigProperty(propertyName = "clientID", propertyValue = "test"),
        @ActivationConfigProperty(propertyName = "maxSessions", propertyValue = "19"),
        @ActivationConfigProperty(propertyName = "acknowledgeMode", propertyValue =
"AUTO_ACKNOWLEDGE"),
        @ActivationConfigProperty(propertyName = "userName", propertyValue =
"anonymous"),
        @ActivationConfigProperty(propertyName = "password", propertyValue =
"anonymous")})
```

<outbound-resourceAdapter> in ra.xml

```
<outbound-resourceadapter>

       <connection-definition>

         <managedconnectionfactory-
class>com.fiorano.mq.ra.outbound.FMQManagedConnectionFactory</managedconnectionfacto
ry-class>

         <connectionfactory-interface>javax.jms.ConnectionFactory</connectionfactory-
interface>

         <connectionfactory-impl-
class>com.fiorano.mq.ra.outbound.FMQConnectionFactory</connectionfactory-impl-class>

         <connection-interface>javax.jms.Connection</connection-interface>

         <connection-impl-class>com.fiorano.mq.ra.outbound.FMQConnection</connection-
impl-class>

       </connection-definition>



       <transaction-support>XATransaction</transaction-support>

       <authentication-mechanism>

           <authentication-mechanism-type>BasicPassword</authentication-mechanism-type>
```

```
          <credential-interface>javax.resource.spi.security.PasswordCredential</credential-
interface>

      </authentication-mechanism>

      <reauthentication-support>false</reauthentication-support>

     </outbound-resourceadapter>
```

Configurations for outbound flow are specified in deployment descriptor 'jms-ds.xml' located in <JBOSS-HOME> /server/default/deploy/jms folder.

Deploying managed connection factory->jms-ds.xml

While deploying the Managed Connection Factory in the J2EE Server, the properties, below, can be specified for the ManagedConnectionFactory:

Type	**Description**	**Property**
ProviderURL	String	Provider URL (where JMS admin objects are stored).
BackupProviderURLs	String	Backup provider URLs, if any.
Initial Context-Factory	String	Name of the class providing the implementation of Initial Context Factory
JndiUserName	String	Username to be used for lookup.
JndiPassword	String	Password to be used for lookup
ConnectionFactory	String	Name of the connection factory to be used for creating a physical connection with the FioranoMQ Server.
ClientID	String	ID to identify the JMS client (message consumer).
JmsUserName	String	Username to be used for creating a connection to the FioranoMQ Server.
JmsPassword	String	Password to be used for creating a connection to the FioranoMQ Server.
XAEnabled	Boolean	Whether or not to enable XA for receiving messages.
Is Transacted	Boolean	Whether or not the session is Transacted. This is used to create a JMS session, if the value of

Type	Description	Property
		the property XAEnabled is false.

Once the connection Factory has been deployed, all the end User needs to do in order to send the message to the FioranoMQ Server is implement standard JMS programming, as shown below:

```
// get JNDI handle

Context jndiCtx = new InitialContext();

// lookup the connection factory

ConnectionFactory cFactory = (ConnectionFactory)jndiCtx.lookup("fmqTCF");

// create connection from factory

Connection connection = connectionFactory.createConnection();

// create session from connection

Session session = connection.createSession(true,AUTO_ACKNOWLEDGE);

// get destination from JNDI

Destination destination = (Destination)session.createTopic("secondaryTopic");

// create a message producer

MessageProducer sender = session.createProducer(destination);

// create a message

TextMessage message = session.createTextMessage();

message.setText(msgData);

// send the message

sender.send(message);
```

29.5.4 Configuring FioranoMQ Resource Adapter in JBoss 4.2.2

Note: JBoss AS 4.2.2 requires JDK1.6 or later versions to function properly.

29.5.4.1 Changes Required for Inbound Communications (Message Inflow Contracts)

The following steps will deploy Fiorano RA and enable Inbound Communication onto JBOSS AS.

1. Shutdown any instance of the JBoss Application Server.

2. Edit ra.xml located at $FIORANO_HOME\fmq\lib\jca\resources\1.4\META-INF and modify the property ProviderURL to point to the appropriate instance of FioranoMQ.
3. Run the buildJcaJars.bat / buildJcaJars.sh script available at FIORANO_HOME\fmq\lib\jca to generate the required .rar files. Two new folders, j2ee1_4 and j2ee1_3 are created. These folders store the generated .rar files.
4. Copy fmq-connector-ra.rar from $FIORANO_HOME\fmq\lib\jca \j2ee1_4 to deploy the directory of the desired JBoss Server configuration profile (all, default, minimal and so on). For example, use $JBOSS_HOME/server/default/deploy.

29.5.4.2 Changes required for Outbound Communication

For EJBs in JBoss to be able to send messages to the FioranoMQ Server, the end User should create Connection Factories. These Connection Factories can be used to create the FioranoMQ connection as per JCA Connection Management contracts.

Connection factories can be created by modifying jms-ds.xml located in $JBOSS_HOME/server/$PROFILE/deploy/jms. The listing below shows the deployment of Sample ConnectionFactories supporting XA and Non-XA connections.

```
<!-Non-XA connection Factory definition.-->

<no-tx-connection-factory>

<jndi-name>FMQNonXA</jndi-name>

<rar-name>fmq-connector-ra.rar</rar-name>

<config-property name="JndiUserName" type="java.lang.String">anonymous</config-
property>

<config-property name="JndiPasswordName" type="java.lang.String">anonymous</config-
property>

<config-property name="ProviderURL"
type="java.lang.String">http://localhost:1856</config-property>

<config-property name="BackupProviderURLs"
type="java.lang.String">http://localhost:1956</config-property>

<connection-definition>javax.jms.ConnectionFactory</connection-definition>

<config-property name="Username" type="java.lang.String">anonymous</config-property>

<config-property name="Password" type="java.lang.String">anonymous</config-property>

<config-property name="ConnectionFactory" type="java.lang.String">primaryTCF</config-
property>

<config-property name="XAEnabled" type="java.lang.Boolean">false</config-property>

<max-pool-size>20</max-pool-size>
```

```
</no-tx-connection-factory>



<!--XA Connection Factory. Use this to use transacted JMS in EJBs-->

<tx-connection-factory>

<jndi-name>FMQXA</jndi-name>

<xa-transaction />

<rar-name>fmq-connector-ra.rar</rar-name>

<connection-definition>javax.jms.ConnectionFactory</connection-definition>

<config-property name="UserName" type="java.lang.String">anonymous</config-property>

<config-property name="Password" type="java.lang.String">anonymous</config-property>

<config-property name="ConnectionFactory" type="java.lang.String">primaryXACF</config-
property>

<config-property name="XAEnabled" type="java.lang.Boolean">true</config-property>

<max-pool-size>20</max-pool-size>

<security-domain-and-application>JmsXARealm</security-domain-and-application>

</tx-connection-factory>

</connection-factories>
```

Note: For a quick start, edit the jms-ds.xml to include the above xml.

Connection factories can be lookedup by J2EE applications for sending and receiving messages from FioranoMQ Topics and Queues.

Restart the JBoss application server.

Sample

FioranoMQ contains samples that demonstrate the usage of FioranoMQ RA. These samples are available in the %FMQ_HOME%\fmq\samples\ApplicationServer\fmqmdb\dist\jboss_4 directory. Refer to *readme_JBoss4.txt* available in the samples\ApplicationServer\ fmqmdb directory for step-by-step instructions on deploying and running the sample.

29.5.5 Configuring FioranoMQ RA in JBoss 4.2.2-XA

To deploy the FioranoMQ Resource Adapter in JBoss 4.2.2, perform the steps below:

1. Stop the instance of JBoss running on the Server. Change the value of the property ProviderURL in *ra.xml* to point to the appropriate instance of FioranoMQ. This .xml file is file located in $FIORANO_HOME\fmq\lib\jca\resources\1.3\META-INF.
2. Run the buildJcaJars.bat/buildJcaJars.sh script available at $FIORANO_HOME\fmq\lib\jca. This will generate the rar files.
3. Copy fmq-connector-ra.rar from $FIORANO_HOME\fmq\lib\jca\j2ee1_3 to deploy the directory of the desired JBoss Server configuration profile (all, default, minimal, and so on). For example, use $JBOSS_HOME/server/default/deploy

   In jms-ds.xml, replace codes appear in bold:

   <!-- The JMS provider loader -->

<mbean code="org.jboss.jms.jndi.JMSProviderLoader"

name="jboss.mq:service=JMSProviderLoader,name=JMSProvider">

<attribute name="ProviderName">DefaultJMSProvider</attribute>

<attribute name="ProviderAdapterClass">

org.jboss.jms.jndi.JNDIProviderAdapter

</attribute>

<!-- The queue connection factory -->

<attribute name="QueueFactoryRef">**java:/XAConnectionFactory**</attribute>

<!-- The topic factory -->

<attribute name="TopicFactoryRef">**java:/XAConnectionFactory**</attribute>

<!-- Uncomment to use HAJNDI to access JMS

<attribute name="Properties">

java.naming.factory.initial=org.jnp.interfaces.NamingContextFactory

java.naming.factory.url.pkgs=org.jboss.naming:org.jnp.interfaces

java.naming.provider.url=localhost:1100

</attribute>

-->

</mbean>

With

<!-- The JMS provider loader -->

<mbean code="org.jboss.jms.jndi.JMSProviderLoader"

name="jboss.mq:service=JMSProviderLoader,name=HAJNDIJMSProvider">

<attribute name="ProviderName">DefaultJMSProvider</attribute>

<attribute name="ProviderAdapterClass">

org.jboss.jms.jndi.JNDIProviderAdapter

</attribute>

<!-- The queue connection factory -->

<attribute name="QueueFactoryRef">**primaryXAQCF**</attribute>

<!-- The topic factory -->

<attribute name="TopicFactoryRef">**primaryXATCF**</attribute>

<!-- Access JMS via HAJNDI -->

<attribute name="Properties">

java.naming.factory.initial=fiorano.jms.runtime.naming.FioranoInitialContextFactory

java.naming.provider.url=http://localhost:1856

</attribute>

</mbean>

4. Start JBoss Server.

Sample

A sample MDB can be run that demonstrates how FioranoMQ RA works with JBoss 4.2.2 (located in the %FMQ_HOME%\JCA\Samples\ directory). Before running this sample, please refer to the *readme_JBoss4.2.2.txt* available in the samples directory for step-by-step instructions on deploying and running this sample.

29.5.6 Configuring MDBs for XA or NONXA

Following configurations should be made in ejb-jar.xml in case of EJB2 or should be specified as annotations in case of EJB3.

Non-XA:

'TransactionType' for MDB should be specified as 'BEAN'

or

'TransactionType' for MDB should be specified as 'CONTAINER' & 'TransactionAttribute' as 'NOT_SUPPORTED'

XA:

'TransactionType' for MDB should be specified as 'CONTAINER'. By default the 'TransactionAttribute' will be 'REQUIRED'.

29.6 FioranoMQ - JBOSS Application Server

FioranoMQ can be integrated with a JBoss application server in which an MDB deployed on JBoss can listen for messages published on FioranoMQ destinations.

This helps in leveraging the power of FioranoMQ as a JMSProvider and Jboss as Application Server.

Refer to the link below to locate the zip file corresponding to JBoss Integration with the FiroanoMQ Server:

http://www.fiorano.com/downloads/fmq/JBossIntegration.zip

This **.zip** file contains the files listed below:

- fmqmdb.jar.
- readme_Jboss4.txt.
- *FMQSenderMDB_1_4.java* (which is the source of the MDB).
- *jms-ds.xml* (which is an .xml file containing sample configurations required for binding the Connection Factories that are to be used).
- jboss-log4j.xml (Which is an .xml file containing suitable appenders for Fiorano logger. This file can be found in location JBOSS_HOME/server/default/conf)

29.6.1 Integrating FioranoMQ with JBoss Application Server 4.2.2

This section describes the steps to integrate FioranoMQ with the JBoss Application Server:

1. Configure FioranoMQ RA in JBOSS 4.2.2. Related instructions has been provided in section 29.5.5 Configuring FioranoMQ RA in JBOSS 4.2.2
2. Navigate to the file jms-ds.xml located at <%JBOSS_DIR%>\server\default\deploy\jms directory (where <%JBOSS_DIR%> is the JBoss 4.2.2 installation directory). A sample of this .xml file is provided in the zip file folder. This file contains Connection Factory bindings which are of vital importance for outbound communication. Modify this file as required.
3. Navigate to the file jboss-log4j.xml located at <%JBOSS_DIR%>\server\default\conf directory (where <%JBOSS_DIR%> is the JBoss 4.2.2 installation directory). A sample of this .xml file is provided in the zip file. This file contains suitable appenders for Fiorano logger. You can change various parameters of this file to set desired log level and logger.
4. Copy the file fmqmdb.jar to %JBOSS_DIR%\server\default\deploy directory.
5. Start the FioranoMQ Server by selecting Start > Programs > Fiorano > FioranoMQ (or by using $FMQ_DIR/fmq/bin/fmq.sh on UNIX).
6. Start the JBoss Server using the startup script run.bat (or run.sh on UNIX).
7. Run a **Publisher** on PrimaryTopic
8. Run a **Subscriber** on SecondaryTopic

Messages can now be sent using the application. These messages are sent on the PrimaryTopic. Messages are received on the JBoss console and are published back to the SecondaryTopic by the MDB, received by the Subscriber.

29.6.2 Integrating FioranoMQ with JBoss Application Server 4.3

Configurations are similar to Jboss AS 4.2.2

jms-ds.xml file is located in <%JBOSS_DIR%>\server\default\deploy

29.6.3 Integrating FioranoMQ with JBoss Application Server 5.1.0

In order to integrate FioranoMQ with JBoss Application Server 5.1.0 follow the steps discussed for integration FioranoMQ with JBoss Application Server 4.2.2. In addition to the steps mentioned, add the entry below to $JBOSS_HOME/server/default/deployers/metadata-deployer-jboss-beans.xml:

```
<entry>

  <key>fmq-connector-ra.rar</key>
  <value><null/></value>
</entry>
```

This element must be added to the Bean <bean name="JBossCustomDeployDUFilter" class="org.jboss.deployers.vfs.spi.structure.helpers.VirtualFileDeploymentUnitFilter">
This element is added by identifying the presence of the preceding element.

```
      <entry>

        <key>jms-ra.rar</key>

        <value><null/></value

    </entry>
```

The EJB2.1 and 3 samples that are described for JBoss 4.2.2 can be used for JBoss 5 as well.

29.6.4 Integrating FioranoMQ with JBoss Application Server 6.1.0

Configurations are similar to Jboss AS 4.3

The EJB2.1 and 3 samples that are described for JBoss 4.2.2 can be used for all above versions.

29.6.5 Using EJB3 compliant MDB while integrating FioranoMQ 9 with JBoss Application Server

The EJB3 compliant MDB (available at http://www.fiorano.com/downloads/fmq/JBoss4.2.2-EJB3.zip) behaves in a manner similar to the EJB2.1 compliant sample described in the section above (and can be deployed using the same steps).

The differences between the two samples are:

- In the EJB3 sample provided, all deployment details have been specified using annotations as part of the java file itself without relying on deployment descriptors. Thus there is no ejb-jar.xml in the jar file.
- The 'destination-jndi-name' parameter has been commented out in jboss.xml while deploying it using annotations. Refer build/META-INF/jboss.xml for more details.
- Also in EJB3, the bean no longer needs to implement the MessageDrivenBean class. Hence instead of the ejbCreate and ejbRemove methods which were earlier used in EJB2.1, we now use PostConstruct and PreDestroy annotations. These two annotations are respectively used to do initialization and cleanup activities in accordance with the beans lifecycle.

Refer to the sample for more details.

Instructions on modifying the EJB3 samples are available in the readme file inside the zip folder.

29.7 FioranoMQ - ATG Dynamo Message Service

A Dynamo Message Service can be made to work with a third party JMS provider. The administrator must declare the provider in the Patch Bay Configuration and must configure the appropriate message sources and sinks to use the Destinations in that provider. Additionally, the client libraries for the JMS Provider need to be in the CLASSPATH of the Dynamo. This can be done by adding the fmq-rtl.jar path to startDynamo.bat, which is the startup script for ATG. The fmq-rtl.jar file can be found in the %FMQ_DIR%\lib directory of the installation package.

29.7.1 Configuring the Dynamo Message System

29.7.1.1 General Architecture

The Patch Bay is a component of the Dynamo Message System that connects messaging components to the JMS Providers. These connections are all defined in the DMS Configuration file, which is an XML file. The Patch Bay is represented in the Nucleus as the component/atg/dynamo/messaging/MessagingManager. The configuration file is generally found at /atg/dynamo/messaging/dynamoMessaging-System.xml. One of the functions of the DMS configuration file is to name all the message sources and sinks existing in the system. Any number of sources and sinks can be specified, and in any order. If there are multiple dynamoMessaging- System.xml files spread across the CONFIGPATH entries, the sources and sinks from all those files are registered. The Patch Bay defines a simple life cycle for message sources and sinks. When the Patch Bay is started, it resolves all the Nucleus names. If the referenced components have not been created, they are created at this time. At this point message sinks must be prepared to receive messages, which can arrive at any time. In some cases message Sources follow a different protocol. After a message source is resolved in the Nucleus, the Patch Bay calls MessageSource.setMessageSourceContext on the component. This provides the component with a context object that can be used to create and send messages. At this point the Patch Bay initializes the various JMS Providers and makes sure that the messaging infrastructure is up and running. It then scans each of the message sources and calls the MessageSource.startMessageSource on each one of them. Message source can start sending messages after this call.

29.7.1.2 Creating Messaging Sources and Sinks

A Message source must implement atg.dms.patchbay. The MessageSource interface and a message sink must implement the atg.dms.patchbay. Sample implementation is available at demomessage.zip. There is an attached *readme.htm* file which contains instructions on running this application. This application uses the localJMS which is a messaging system provided by Dynamo.

To register the application as a Nucleus Component, follow the steps below:

1. Navigate to %DYNAMO_HOME%\bin.
2. Enter startDynamo - m demomessage (start Dynamo.bat is a batch file used to start ATG). This starts the server and registers **demomessage** as a Nucleus component, while the localJMS (FioranoMQ in this case) remains its messaging backbone.

To configure the system to use FioranoMQ for messaging follow the steps below:

1. Write a simple class that implements the interface atg.dms.patchbay.JMSInitialContextFactory, which defines a single method createInitialContext. *FioranoImpl.class* is an implementation of this simple class. This class should be placed in %DYNAMO_home%\locallib, which, by default, is available in the startup classpath. If the package is custom-made, add the absolute path to startDynamo.bat, located in %DYNAMO_home%\bin.

2. Register this class as a Nucleus Component. To register either an ATG Control Center can be used or a simple properties file may be used and saved in the directory %DYNAMO_home%\localconfig. If this file is located elsewhere, the CONFIGPATH in postEnvironment.bat must be changed. FioranoImpl.properties is a sample of the aforementioned file. This file can be saved in the specified directory. The name of this file has to match the name of the class file in which the interface atg.dms.patchbay.JMSInitialContextFactory is implemented.
3. Replace the dynamoMessagingSystem.xml, located in **demomessage\config\atg\dynamo\messaging** with the one that is available.

The description of the integration here is based, in part, on the work of Stuart Jones.

29.8 FioranoMQ - Oracle Weblogic Application Server 9.0

High-performance Message Driven Beans (MDBs) can be developed through the integration of FioranoMQ Server with the BEA WebLogic Server. This section explains the steps required to integrate FioranoMQ Server with the BEA WebLogic 9.2 Application Server.

29.8.1 Integrating FioranoMQ with Oracle Weblogic Application Server

The steps, below, are required to integrate the FioranoMQ Server with the BEA Weblogic Server:

1. Install Weblogic Platform 9.2.3. Copy %FMQ_DIR%\lib\client\all\fmq-client.jar to %WL_HOME%\server\lib.
2. Include %WL_HOME%\server\lib\fmq-client.jar in the CLASSPATH of %WL_HOME%\samples\domains\wl_server\setExamplesEnv.cmd. In addition, set WL_HOME = path of your Weblogic home directory.
3. Navigate to the folder %WL_HOME%\samples\server\src\examples\ejb20\message and copy the files MessageTraderBean.java, Client.java, ejb-jar.xml and weblogic-ejb-jar.xml extracted from the attached zip file WeblogicIntegration.zip.
4. Run %WL_HOME%\samples\server\config\examples\setExamplesEnv.cmd on the command line. This provides the environment required to run the sample.
5. Navigate to the folder %WL_HOME%\samples\server\examples\src\examples\ejb\ejb20\message through command prompt and execute the file build.cmd present in the attached zip file WeblogicIntegration.zip. This compiles the Client.java file and adds it to the folder %WL_HOME%\samples\server\examples\build\clientclasses\examples\ejb20\message . In addition, this rebuilds ejb20_message.jar and adds it to the %WL_HOME%\samples\server\examples\src\examples\ejb\ejb20\message\dist.
6. Navigate to the folder %WL_HOME%\server\bin and append the CLASSPATH of fmq-client.jar to the existing CLASSPATH found in the file startWLS.cmd %WL_HOME%\server\lib\fmq-client.jar.
7. Start the Weblogic Examples Server by clicking on Start > Programs > BEAWeblogic Platform 9.2.3 > Weblogic Server 9.2.3 > Server Tour and Examples> Launch Examples Server.
8. Start the FioranoMQ Server by clicking on Start > Programs > Fiorano > FioranoMQ> FioranoMQ Server.

9. Run java examples.ejb25.message.Client http://<FMQ_HOST>\<FMQ_PORT>. By default, the FMQ_PORT is 1856. The Client.java sends three messages on primaryTopic located on the FioranoMQ Server.

   On the console window of Weblogic Example Server, messages are displayed indicating that messages were received by the MDB deployed on the Weblogic server, listening on primaryTopic and located on FioranoMQ. Download Necessary Files.

29.8.2 Troubleshooting

29.8.2.1 Setting up the Bean as a Durable Subscriber with ClientID and Subscriber ID

One commonly encountered problem is that although the clientID is specified in the weblogic-jar.xml, the FioranoMQ server admin console accepts it as the subscriberID (client name). This means that the clientID in weblogic-jar.xml becomes the SubscriberID for the FioranoMQ Server. In Weblogic creating the ConnectionFactory or setting its parameters (ClientID) is not done through weblogic-ejbjar.xml. Weblogic-ejbjar.xml is used only to lookup the ConnectionFactory. Therefore, the ClientID cannot be set through weblogic-jar.xml.

As per to JMS settings: The preferred way to assign the client identifier of a client is to configure a client specific ConnectionFactory and then to transparently connect to the connection it creates. Alternatively, a client can set the client identifier of a connection using a provider-specific value.

The ClientID can be set either on the ConnectionFactory or on the Connection. Therefore, a ClientID can be set while making a new ConnectionFactory. In the default ConnectionFactories, the ClientID is set to a particular default value.

Use Connection.setClientID() to set a ClientID on a Connection manually.

29.8.2.2 IncompatibleClassChangeError

The error IncompatibleClassChangeError is thrown when an incompatible class change occurs. The definition of one of the classes, on which the currently executing method depends, has since changed. This issue can be resolved by adding the fmq-client.jar path in the CLASSPATH.

29.9 FioranoMQ - Borland Enterprise Server 5.1

It is possible to seamlessly integrate FioranoMQ with the Borland Enterprise Server (BES).

Below are the steps required to integrate FioranoMQ with the Borland Enterprise Server:

1. Install BES 5.1. A developer version of Borland Enterprise Server 5.1 is available at http://www.borland.com
2. Copy fmq-client.jar from %FMQ_DIR%\lib\client\all directory to %BES_HOME%\lib directory of the BES installation package.
3. Add the jndi.properties file to the fmq-client.jar present in the library directory of the BES installation package.

4. Replace jndi.properties file in %BES_HOME%\lib\asrt.jar with the jndi.properties.
5. Start the FioranoMQ server by selecting Start > Programs > Fiorano > FioranoMQ> FioranoMQ Server (%FMQ_DIR%\bin\fmq.bat)
6. Start the BES server by clicking on Start > Programs > Borland Enterprise Server > Server.

29.9.1 Deploying of FioranoMQ Libraries

To integrate FioranoMQ with Borland, deploy FioranoMQ libraries to a BES partition.

To deploy the required libraries, follow the steps below:

1. Start the BES console by clicking on Start > Programs > Borland Enterprise Server > console. The default username and password are admin and admin, respectively.
2. Click on **Wizards** to open the deployment wizard.
3. Add %BES_HOME%\lib\fmq-client.jar to the modules.
4. Ensure that **Restart Partitions** on **Deploy** is checked. Click on **Next**.
5. Add it to the standard partition and click on **Finish**.

FioranoMQ libraries are deployed successfully and are seen in the deployed components of standard partition.

29.9.2 Environment Variables

Environment Variables should be appropriately set and all environment variables should follow the steps given below:

1. Appclient

Add FioranoMQ libraries to the Appclient tool to run the samples. The CLASSPATH must first be appended in %BES_HOME%\bin\appclient.config, by adding the following lines:

addpath $var(installRoot)/fmq\lib\client\all\fmq-client.jar

2. Environment

Create a new batch file named setenv.bat (setenv.sh on UNIX) in %BES_HOME% with the lines:

1. Set BES_HOME=c:\bes (replace it with your home directory)
2. Set VBROKER_ADM=%BES_HOME%\var\servers\aseem\adm
3. Set VBROKERDIR=%BES_HOME%
4. Set OSAGENT_PORT=14000
5. Set PATH=%PATH%;%BES_HOME%\bin;%BES_HOME%\jdk\bin

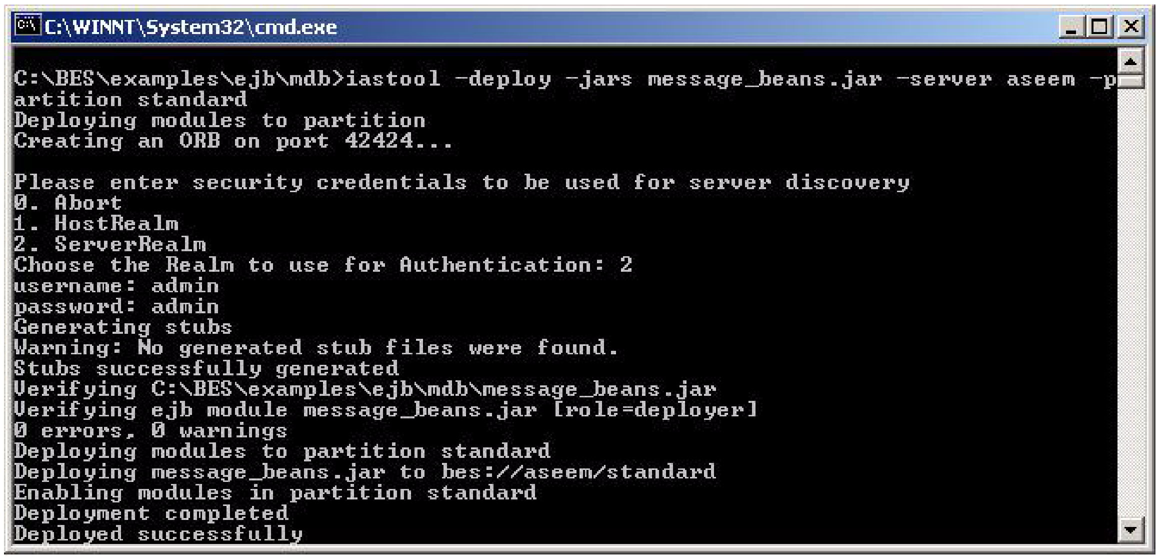
29.9.3 Samples

To run JMS samples, follow the steps below:

1. Integrate FioranoMQ with the Borland Enterprise Server and deploy FioranoMQ libraries to the BES partition.
2. Set all the environment variables.
3. Unzip the attached MQ sample and navigate to the directory the attached sample is extracted.
4. Deploy the message_beans.jar in the server using the following code:

iastool -deploy -jars message_beans.jar -server aseem -partition standard

Choose **option 2** (serverrealm). The default username is admin, and default password is admin. The output appears as shown in **Figure 12-3**:



```
C:\BES\examples\ejb\mdb>iastool -deploy -jars message_beans.jar -server aseem -p
artition standard
Deploying modules to partition
Creating an ORB on port 42424...

Please enter security credentials to be used for server discovery
0. Abort
1. HostRealm
2. ServerRealm
Choose the Realm to use for Authentication: 2
username: admin
password: admin
Generating stubs
Warning: No generated stub files were found.
Stubs successfully generated
Verifying C:\BES\examples\ejb\mdb\message_beans.jar
Verifying ejb module message_beans.jar [role=deployer]
0 errors, 0 warnings
Deploying modules to partition standard
Deploying message_beans.jar to bes://aseem/standard
Enabling modules in partition standard
Deployment completed
Deployed successfully
```

Figure: Output

5. Run the client using the following code:

appclient message_beans_client.jar

The output, shown below, is displayed:

Client end:

Sending a message to queue PrimaryQueue.

Publishing a message to topic primaryTopic.

Done.

EJB container log messages (in the event option of log tab):

<HelloEJBQueue> Got a message from queue primaryQueue:

Hello MDB, this is a message from the client...

<HelloEJBTopic> Got a message from topic primaryTopic:

Hello MDB, this is another message from the client...

29.10 FioranoMQ - Orion Application Server

FioranoMQ can be integrated with Orion. Execute the following steps to integrate the Orion Application Server with FioranoMQ:

Note: Since Oracle 9iAS has licensed Orion, the procedure, below, should also work well with it.

While starting up Orion the fmq-rtl.jar (located in the /fmq/lib directory of the FioranoMQ installation package) needs to be present in the CLASSPATH.

1. When integrating an external JMS system with Orion, the implementation of an interface defined by Orion called ResourceProvider is provided by the external JMS system. A default implementation of the interface is provided by Orion that allows a User to plug in a JMS that is JNDI enabled. FioranoMQ is JNDI enabled but the current default implementation of the ResourceProvider interface is for an older version of the Resource Provider and may not work. FioranoMQ provides its own implementation of the ResourceProvider interface called ContextScanningResourceProvider.class. This file replaces the default file available in the orion.jar library.
2. Modify the application.xml located in %Orion_HOME%\config directory. This .xml defines the implementation of the Resource Provider interface that is used by the Orion Server.
3. The file server.xml defines the configuration parameters for the Orion Server. Information about any Application/Bean that is to be loaded or deployed at startup is given in this file. In FioranoMQ, this application is a simple MDB with auto-start=**true**. The Orion Server looks for the ear file of the Application/Bean along the path provided. The correct path of the file mdb.ear must to be added to the server.xml file.

Note:

- The portion of the JMS server located in server.xml at %Orion_HOME%\config needs to be commented out to prevent the startup of Orion's JMS Server.
- jndi.properties: This file holds parameters that are needed for loading Connection Factories and is available at mdb_hello_world.zip
- Unzip the complete application (mdb_hello_world.zip) located at %ORION_HOME%. [Link?] Replace the server.xml and application.xml in %Orion_HOME%\config and then run the application. The file Main.java is a sample client that sends a single message on FioranoMQ. The onMessage() method of MDB is invoked on receipt of this message.
- Specify the **ack** mode of the session created by the MDB. If not specified, the ack mode results in -1, which is not allowed. To change this the following modifications, below, must be made to the *ejb-jar.xml*:

```
<transaction-type>Bean</transaction-type>

<acknowledge-mode>Auto-acknowledge</acknowledge-mode>
```

If problems are encountered, please contact techsupport@fiorano.com

29.11 FioranoMQ - IBM WebSphere Application Server 5.1

FioranoMQ can be integrated with a WebSphere Application Server so that MDBs deployed on WebSphere can listen in for messages published on destinations that exist on the FioranoMQ Server.

This document provides instructions detailing the integration of FioranoMQ with a Web Sphere Application Server.

A sample code is provided in the attached zip file.

29.11.1 Assumptions

In the section below, the assumptions, below, have been made:

The Web Sphere Application Server and the Application Developer are installed in the \wsad directory. In case you have installed them in a different directory, please make the necessary changes.

FioranoMQ is installed in the \Program Files\Fiorano\FMQ directory. In case you have installed it in a different directory, please make the necessary changes.

29.11.2 Configuring WS Application Developer for FioranoMQ

The remaining sections in this document focus on the steps that need to be carried-out to configure the WebSphere Application Server via the WS Application Developer Environment. Instructions to configure the WebSphere Application Server via the WS Application Developer Environment are divided into the following sections:

1. Define the WebSphere Test Server
2. Configure WebSphere to be able to use FioranoMQ as a JMS provider
3. Create/Configure MDB in the WebSphere Application Developer Environment
4. Deploy/Test the MDB

29.11.2.1 Define a WebSphere Test Server

Below are the instructions to create and configure a WS Server to enable it to run an EJB component:

1. Open a Server perspective in Application Developer.
2. Right-click on the Navigator view and select **New** > **Server Project**.
3. Type **ServerProject** in the Project name field and click **Finish**.
4. Right-click on **ServerProject** and select New > Server and Server Configuration.
5. In the Server name field, type **TestServer**.

6. Set the Server type as WebSphere 5.1 Test Environment.
7. Click **Finish**.
8. In the Server Configuration view, double-click **TestServer** under **Server Configuration**s.
9. Under the **Configuration** tab, check **Enable administration client**.
10. Save the configuration by pressing Ctrl-S.
11. In the **Servers** view, right-click on the **TestServer** and select **Start**.
12. Wait until you see the message **server1 open for e-business** on the console before proceeding.

29.11.2.2 Configure WS to use FioranoMQ as JMS provider

EJB components can use the administered objects (Connection Factories and Destinations) residing on the FioranoMQ server.

Steps to define FioranoMQ as the JMS provider:

1. In the **Servers view**, right-click on **TestServer** and select **Run administrative client**.
2. At the **Login** screen, click **Submit**.
3. Click the **+** sign next to **Resources**.
4. Click **Generic JMS Providers**.
5. Click **New**.
6. In the Server Class Path field add the entries, separated by a semicolon, on a single line in the Server Class Path text field:

   <FMQ_HOME>\lib\fmq-rtl.jar

   (Where FMQ_HOME refers to the installation directory of FioranoMQ. This makes the FioranoMQ Java classes available to the server.)
7. Change the **Name: field** to **FioranoMQ**.
8. Change the **Description: field** to **FioranoMQ** as the JMS Provider for IBM Web-Sphere Application Server
9. Change the **External Initial Context Factory: field** to **fiorano.jms.runtime.naming. FioranoInitialContextFactory**.
10. Change **the External Provider URL: field** to **http://localhost:1856**
11. Click **OK** and then save the changes.

Steps to configure the JMS Destination under FioranoMQ resource:

8. Click the **FioranoMQ** link under **Generic JMS Providers,** and click **JMS Destinations.**
9. Click **New**. In the **Name**: field, type **MyQueue**.
10. Change the JNDI **Name: field** to **jms/MyQueue**.

11. Set the **Destination Type field** to **QUEUE.**
12. In the External **JNDI Name field**, type **primaryQueue** and click **OK**.
13. Go to the **FioranoMQ** link under **Generic JMS Providers**.
14. Click **JMS Connection Factories**.
15. Click **New**. In the **Name: field** type **MyQCF**.
16. Change the **JNDI Name: field** to **jms/MyQCF**.
17. Change the **External JNDI Name: field** to **primaryQCF**.
18. Set the **Connection Type: field** to **QUEUE** and click **OK.**
19. Click on **Save** link at the top of the page to save all changes to the master configuration.

29.11.2.3 Create/Configure MDB in WebSphere application developer

MDB can be created by following the steps below:

Create the new EJB Project

1. Switch to the **J2EE navigator** view in the **J2EE perspective**.
2. Start creating a new project by selecting **File > New > EJB Project**. This opens the EJB Project Creation wizard. Check that a 2.0 EJB Project is in the process of being created and click **Next**.
3. Specify the **Project name** as **MDB** and **EAR Project** to as **FioranoEAR**. Click on **Finish.** The **Application Developer** creates the new project.

Create the bean class

1. In the J2EE Hierarchy view **right-click** on the new MDB EJB module and select **New > Enterprise Bean**. This opens the **Enterprise Bean Creation wizard**.
2. Check that the specified EJB project is **MDB.** Click **Next**. The **bean type** and **Properties panel** is displayed.
3. Specify the type of message-driven Bean. The Bean name is QReceiver and the default package is named fiorano.jms. Click **Next**.
4. Specify the transaction as **container** and the destination as **Queue**. Set the listenerPort to ReceiverListenerPort. Click **Finish.** The **Application Developer** creates the **Receiver MDB**.

Note: For further details refer to the section on **Configure Message Listener Service**

Complete the Bean Class

Once the Application Developer has created the **Receiver MDB**, a new entry called **QReceiver** appears under the MDB EJB module in the J2EE hierarchy view.

1. Expand the **QReceiver** entry and double-click the **QReceiver-Bean** to open its editor.
2. In the **QReceiverBean** editor there are a series of methods that are defined by the **Application Developer** upon the creation of the **Receiver MDB**.

3. Modify the Bean's onMessage () method to print the message on the console. Save the changes. (Modified QReceiver-Bean.java is also provided with this document for reference.)

Configure the Message Listener Service

The User must configure the destination on which the bean listens for published messages. To configure this destination, follow the instructions below:

1. **Log in** to the administrative console.
2. Expand the Servers entry in the left-hand navigation frame and click **Application Servers.**
3. In the frame that is displayed, click on **server1** and then **Message Listener Service** in the **Additional Properties** table. This opens the message listener configuration frame.
4. Click **Listener Ports**. Click on **New** to create a new **Listener Port** for the **Receiver MDB**.
5. Specify the name of the port as **ReceiverListenerPort** and the Connection Factory name as jms/MyQCF. The destination name is jms/MyQueue. Note: These names match the Connection Factory name and Destination name used while configuring the WS Server to use FioranoMQ as a JMS provider.
6. Click on **OK**. Click the **Save** button to save changes.
7. Close the **administrative** console. The Receiver MDB is now ready for deployment.

29.11.2.4 Deploy/Test the MDB

Deploy MDB

To deploy an MDB, follow the instructions given below:

1. Start the FioranoMQ Server on the same machine as the WSAD. (**Start>Programs> Fiorano>FioranoMQ>FioranoMQ Server**).
2. **Right-click** the **Receiver MDB** entry in the J2EE Hierarchy. Click **Run** on the Server.
3. In the server selection window, choose the existing server as **TestServer**. Click **Next**.
4. Deploy EJB Beans and click **Finish**.

If the Test Server was running, it will restart after deployment.

Test the MDB

The MDB begins listening for messages on **jms/MyQueue** upon deployment. So a message is pushed onto the **primaryQueue** (for example, via a Sender sample) then the message will show up in the console window of the **Application Developer** as **Test Server.**

1. **Launch** the FioranoMQ Application Console (by clicking on **Start > Programs > Fiorano > FioranoMQ > FioranoMQ Console**).
2. Run a Sender Application (present in the fmq\samples\Ptp\SendReceive directory) using the command below:
   run-client QSender

3. Send a message. The console view of the WS Application Developer will display:

```
[2/17/04 12:13:06:188 IST] 75ed4cbf SystemOut O Bean got message from onMessage():

                hi



[2/17/04 12:13:09:812 IST] 75e04cbf SystemOut O Bean got message from onMessage():

                How are you?
```

29.12 FioranoMQ - IPlanet Application Server 7

This section describes the deployment of the Message Driven Beans (MDBs) on the IPlanet Application Server (now known as SunOne Application Server) through the integration of FioranoMQ with the IPlanet Application Server. The steps required to integrate FioranoMQ with SunOne Application Server 7 are explained in this section.

29.12.1 Installing and Setting up the Iplanet Application Server 7

Follow the steps below to install the IPlanet Application Server 7:

1. Expand or unzip the installation bundle and save to a temporary directory.
2. Navigate to the directory in which the files are saved and double-click the setup.exe file.
3. Read the **Welcome screen** and click **Next.**
4. Enter the path to the installation directory or click the ellipsis (...) to browse for a directory.
5. In the Server Configuration Information dialog box, enter the information given below:
    a. Admin User: Name of the User who administers the Server.
    b. Administration User's Password: Password to access the Administration Server. Minimum number of characters is 8. Re-enter the password in the text box to confirm the password.
    c. Administration Server Port: Port number to access the Administration Server. A default port number will be displayed (for example 4848). Change this default port number, if necessary **Click Next**. The installation program checks port numbers for validity and availability.
    d. HTTP Server Port: Port number to access the initial Server instance.
6. When the installation process is complete, the **Installation Complete** screen will be displayed. Review the content of this screen and click **Finish** to exit the installer.
7. Set the PATH environment variable: Add the value of the <install_dir>/bin to the front of the PATH value. For example, add **\Sun\AppServer7\bin**; to the front of the variable value through the **Control Panel**.

Troubleshooting Environment Settings

The following result is obtained upon execution of the asadmin command:

```
C:\>asadmin

Use "exit" to exit and "help" for online help

asadmin>
```

This indicates that the environment settings have been configured successfully.

Starting Administrative Server and Application Server

SunOne Application Server 7 introduces a feature named Administrative Domains that enables defining multiple, disparate Application Server runtime configurations that reuse the same installation image. Each Administrative Domain is represented by an Administrative Server, which in turn controls one or more application Server instances. The configuration of an Administrative Domain can reside anywhere on the machine.

Use the following command to start both the Administrative Server and the Application Server instances:

```
C:\>asadmin start-domain --domain domain1
```

In the above command, **domain1** is the name of the Administrative Domain defined by default during the installation of the IPlanet Application Server.

The following result is obtained after the command is executed:

```
asadmin start-domain --domain domain1

Instance domain1:admin-server started

Instance domain1:server1 started.

Domain domain1 Started.
```

29.12.2 Configuring FioranoMQ Server for IPlanet Integration

Below are the steps required to configure the FioranoMQ Server for IPlanet integration:

1. Start the FioranoMQ Server (**Start>Programs> Fiorano>FioranoMQ>FioranoMQ Server**).
2. Create JMS_MYQCF as a QueueConnectionFactory on FioranoMQ.
3. Create JMS_MYQUEUE as a Queue on the FioranoMQ Server.

29.12.3 Configuring IPlanet Application Server 7 for the FioranoMQ Server

Below are the steps required to configure IPlanet Application Server 7 for the FioranoMQ Server:

1. Using the web browser access the SunOne Administration Tool at **http://<host>:<admin_port>.** (The <host> is the machine on which the application server is installed and the <admin_port> is the server administration port (4848 by default).
2. From the left frame click **AppServer Instances > server1**.
3. From the right frame select **JVM Settings** tab.
4. Click on **Path Settings**.
5. Enter the following files in the CLASSPATH text box:

   On Windows:

   <fiorano_install_dir>/fmq/lib

<fiorano_install_dir>/fmq/lib/fmq-rtl.jar

   (Where fiorano_install_dir is the directory in which the FioranoMQ Server is installed.)
6. In the Native Library Path Suffix text box, type:

   <fiorano_install_dir>/fmq/lib
7. Click **Save**.

To transfer the above modifications to the Server, follow the steps below:

1. Select the **General** tab and click on **Apply Changes.**
2. Stop and re-start the Server instance.
3. Navigate to the folder <appserver_install_dir>\samples\ejb\mdb\simple\src. In the file sun-application-client.xml, change the tag value of tags <resref-name> and <jndi-name> under tag <resource-ref>to jms_MyQcf.
4. Change the value of tags <resource-env-ref-name> and <jndiname> under tag <resource-env-ref> to jms_MyQueue. FioranoMQ has a file based data store and creates folders based on the name of Queue. Folders cannot have names containing characters '/', '\'. '*', '?', '<', '>', '|'. Similarly, in the file sun-ejb-jar.xml, the value of tag <jndi-name>, under the tag <ejb> must be changed to jms_MyQueue and the value of tag <jndi-name>, under the tag <mdb-connection-factory> must be changed to jms_MyQcf.
5. In the file applicationclient.xml, change the value of tag <res-ref-name>, under tag <resourceref> to jms_MyQcf and the value of tag <resource-env-ref-name>, under tag <resource-env-ref> to jms_MyQueue.
6. Navigate to the folder <**appserver_install_dir>\samples\ejb\mdb\simple\src\samples\ejb\mdb\simple\client**. In the file **SimpleMessageClient.java**, change the lookup names to jms_MyQcf instead of java:comp/env/jms/MyQcf and jms_MyQueue instead of java:comp/env/jms/MyQueue.

29.12.4 Registering JMS Resources with IPlanet Application Server 7

To register the JMS Resources with IPlanet Application Server 7 follow the steps below:

1. Run the asadmin utility in multimode. For example: <AS_HOME>/bin/asadmin multimode. (Where <AS_HOME> is the SunOne Application Server installation home.)
2. Set global parameters so that you do not have to enter them for every sub command. For example:

   asadmin>export AS_ADMIN_USER=<admin_username> AS_ADMIN_PASSWORD=<admin_password> AS_ADMIN_HOST=<host> AS_ADMIN_PORT=<adminserver_port> AS_ADMIN_INSTANCE=<instance_name>

   In the above code:

   <admin_username> is the Application Server administration User name.

   <admin_password> is the administration password.

   <host> is the machine on which the application Server is installed.

   <adminserver_port> is the administration Server port of the Application Server.

   <instance_name> is the Application Server instance name (example, server1).
3. Create external jndi external resources.

For the MDB sample application, create one external jndi resource with the resource type of javax.jms.QueueConnectionFactory and one external resource with the resource type of javax.jms.Queue. Refer to the code below:

asadmin>create-jndi-resource --jndilookupname

<qcf_jndi_lookup-name> --resourcetype

javax.jms.QueueConnectionFactory --factoryclass

fiorano.jms.runtime.naming.FioranoInitialContextFactory --enabled=true --property

java.naming.provider.url=<directory_path>:java.naming.security.authentication=none

<qcf_jndi_name>

asadmin>create-jndi-resource --jndilookupname

<q_jndi_lookup-name> --resourcetype javax.jms.Queue --factoryclass

fiorano.jms.runtime.naming.FioranoInitialContextFactory --enabled=true --property

java.naming.provider.url=<directory_path> <q_jndi_name>

In the above code,

- <qcf_jndi_lookup-name> and <qcf_jndi_name> should be replaced by jms_MyQcf.

- <q_jndi_lookup-name> and <q_jndi_name> should be replaced by jms_MyQueue
- <directory_path> is the providerURL i.e. http\://localhost\:1856 for the FioranoMQ Server.

The escape (****) character has been used in front of the colon (**:**) in the directory path.

4. List the external jndi resources to verify the resources created. For example:

   asadmin>list-jndi-resources –-user <admin_username e.g. admin> --password

   <admin_password> --host <where appserver is running e.g. localhost> –-port 4848

   <server_instance_name e.g. server1>

5. Reconfigure the Server instance to make sure that the changes made have been transferred to the Server. For example:

   asadmin>reconfig instance_name

29.12.5 Compiling and Deploying the Sample mdb-simple.ear

Remove **guest** from <name> element and <password> element and leave these elements blank <appserver_install_dir>\samples\ejb\mdb\simple\src\sunapplication-client.xml. appserver_install_dir is the directory in which IPlanet Application Server 7 (SunOne AppServer) has been installed.

1. In the command prompt, navigate to the src directory of the sample. For example:

<appserver_install_dir>\samples\ejb\mdb\simple\src

2. Execute the asant command. It recompiles the mdb and recreates the ear file mdb-simple.ear.
3. Execute the asant deploy command. It causes the mdb-simple.ear to deploy on the SunOne AppServer.

29.12.6 Running the Sample Application

1. Navigate to the <appserver_install-dir>/bin directory and edit the file appclient.bat. Add <fiorano_install_dir>\fmq\lib\fmq-rtl.jar;<fiorano_install_dir>\fmq\lib; to JVM_CLASSPATH.

   For example,

   Add c:\Progra~1\Fiorano\Fioran~1\fmq\lib\fmq-rtl.jar;\Progra~1\Fiorano\Fioran~1\fmq\lib;

   To

   JVM_CLASSPATH. Add <fiorano_install_dir>\fmq\lib

   For example,

   Add C:\Progra~1\Fiorano\Fioran~1\fmq\lib;

To

LD_LIBRARY_PATH

cd <appserver_ install_dir>/domains/domain1/server1/applications/j2ee-apps/mdb-simple_1

2. Execute the command > appclient -client mdb-simpleClient.jar -name Simple-MessageClient - textauth

The following output is displayed:

```
Sending message: This is message 1

Sending message: This is message 2

Sending message: This is message 3
```

The Message Bean receives the messages and they are printed in the log file (<appServer_install_dir>\domains\domain1\server1\logs server.log):

```
MESSAGE BEAN: Message received: This is message 1

MESSAGE BEAN: Message received: This is message 2

MESSAGE BEAN: Message receive: This is message 3
```

29.13 FioranoMQ - OC4J Application Server

FioranoMQ can be integrated with an OC4j Application Server. The sample code that is provided in the following zip file can be downloaded from ftp://downloads.fiorano.com/fmqdownload/OC4JIntegration.zip

29.13.1 Deploy the MDB Application

Download the MDB application from:

ftp://downloads.fiorano.com/fmqdownload/fiorano.ear

Save the MDB Application on the local drive. in the \fiorano\MDB directory.

Follow the steps below to deploy MDB on OC4J:

Add the following lines of code [Which code? Specify please] to the *server.xml* file which can be located at \oracle10g\j2ee\home\config directory

Note: It is assumed that the Oracle 10g Application Server is installed at the \oracle10g directory.

```
<application name="mdb" path=" c:\fiorano\MDB \fiorano.ear" auto-start="true"/>

<library path=\Program Files\Fiorano\FMQ\fmq\lib\fmq-rtl.jar" />
```

The file **server.xml** defines the configuration parameters for the OC4J Server. This file also contains information regarding any Application or any Bean requiring loading or deploying at startup. InFioranoMQ the Application is a simple MDB with auto-start set to **true**. The OC4J Server looks for the ear file of the Application or of the Bean along the path provided by the application name. The correct path of the file mdb.ear needs to be added to the *server.xml* file.

Start the OC4J Server by entering the command below in

\oracle10g\j2ee\home directory:java -jar oc4j.jar

This starts the 0C4J Server and deploys the MDB on it.

29.13.2 Test the MDB

Upon deployment, the MDB starts listening for messages on the **primaryQueue**. If a message is pushed onto the **primaryQueue** (via a Sender sample for example), the message should show up in the output logs of the Oracle Application Server.

1. Run a sender application (present in fmq/samples/Ptp/SendReceive directory) using the following command:

   run-client QSender

2. Now send a message. This message will appear in the OC4J console as shown below:

   04/06/28 13:32:01 Got
   Message=fiorano.jms.services.msg.def.FioranoTextMessage@354749

29.14 FioranoMQ – Sun GlassFish Enterprise v2.1

FioranoMQ can be integrated with Sun's GlassFish Application Server. The sample MDB (Message Driven Bean) can be downloaded from: http://www.fiorano.com/downloads/fmq/GlassFishIntegration.zip

The .zip file included in the download contains the following files:

- fmqmdb.jar.
- readme.txt.
- A source file; FMQSenderMDB_1_4.java (which is the source of the MDB).

To observe the MDB, follow the steps below:

1. Start the FioranoMQ Server.
2. Run the Application Server as admin start-domains domain1
3. Deploy the FioranoMQ Resource Adapater (fmq-connector-ra.rar) as a connector module. This can be done through the command line or by logging into the administration console at http://<ip>:4848. The connector module is available in the side pane of the frame under Applications.

Note: As mentioned at the beginning of this chapter, the resource adapter has to be configured before deployment. By default the resource adapter assumes that the FMQ Server is running at http://localhost:1856. Properties can be modified as per the requirement before deploying the resource adapter.

4. Configure Connector-connection-pools (available under Resources -> Connectors in the side pane of the admin console) and Connector-resources (available under Resources -> Connectors) based on whether outbound communication from MDB is necessary.

   The sample MDB demonstrated here requires that there be a Non-Transaction based Connection Factory named FMQNonXA which will be used to lookup its target destination upon which it will publish the messages received from FioranoMQ.

5. Create a new connector-connection-pool named 'outpool' and select the resource adapter as 'fmq-connector-ra' and connection definition as 'javax.jms.ConnectionFactory'. Set transaction support to 'No Transaction' in the next page as we require a non transaction based connection factory.

6. Create a new connector resource named FMQNonXA which uses the 'outpool' connection pool created.

   Note: These two resources can also be configured/created through the command line.

7. Deploy the sample MDB fmqmdb.jar (Applications → EJB Applications → Deploy).

8. Start a FioranoMQ publisher on the primaryTopic and a subscriber on the secondaryTopic. The MDB reroutes all messages from the primaryTopic to the secondaryTopic.

29.15 FioranoMQ – Apache Tomcat integration

FioranoMQ can be integrated with Apache Tomcat application server. The sample and Instruction related to it can be downloaded from http://www.fiorano.com/downloads/fmq/FMQTomcatIntegration.zip

The .zip file contains the following files:

- Sample Web application to send and receive messages from FMQ server in “FMQSample” folder.
- Resource descriptor file “FMQSample.xml”.
- Instructions about Tomcat Integration in “TomcatIntegration.pdf”.

29.16 To Run JMS Java Applications

To run JMS Java Applications:

Include the jar files, below, in the application CLASSPATH:

- extlib/jms/jms.jar [not required with JavaEE edition 1.4 and above]
- fmq/lib/client/all/fmp-client.jar
- framework/lib/all/fiorano-framework.jar

In addition, include the jar files, below, for HTTP or HTTPs transport:

- extlib/httpclient/httpclient.jar
- extlib/sslava/sslava.jar
- extlib/sslava/sslava_1.2.3_patch_5.jar

Note: All the paths are relative to the Fiorano installation directory.

Chapter 30: Create Custom MBean Service

30.1 Creating Custom MBean Service for the FioranoMQ Server

Below are the steps to create and register a custom MBean Service for the FioranoMQ Server:

1. **Create Service:** (See Sample 1.)
    a. Create an MBean Java Class and extend the fiorano.jms.jmx.ComponentMBean.
    b. Override life cycle methods including createService(), startService() and stopService() which are invoked by the Fiorano Server during server initialization.
    c. Write the code specific to the service in these methods. (Look at the sample 1.)
    d. Compile using compile-client.bat.
    e. Write an .xml file of the same name describing the MBean's attributes and save it in the same directory as the MBean class file. Create a jar file. (See Sample 4.)Update the fmq.conf with the location of this jar file (under the java.classpath heading).
    f. To install FioranoMQ as a Windows NT Service, update %FIORANO_HOME%/WinService/conf/fmq.conf with the location of this jar file.

Sample 1:

```
package com.customMBean.mq.fiorano;



import fiorano.jms.jmx.ComponentMBean;

import java.io.PrintStream;

import fiorano.jms.common.FioranoException;

import fiorano.jms.route.def.*;



public class PubSubTestAdapterMBean extends ComponentMBean

{

   public PubSubTestAdapterMBean()

   {

   }
```

```
public void createService()

{

    System.err.println("creating PubSubTest...");

    pubsubtest = new PubSubTest();

    pubsubtest.printConfiguration();

}


public void startService()

{

    System.err.println("starting PubSubTest...");

    pubsubtest.run();

}


public void stopService()

{

    System.err.println("stopping PubSubTest...");

}


public void destroyService()

{

    System.err.println("destroying PubSubTest...");

}


public Object getResource()

{

    return this;
    }



        public void setFMQConnectionManager(javax.management.ObjectName obj){

        }



    protected PubSubTest pubsubtest;

}
```

2. **Configure Service:** (See Sample 2.) [Link?]
    a. Create a directory in the %FIORANO_INSTALL_DIR%\fmq\profiles\FioranoMQ\deploy\services.
    b. For example- Create a directory PubSubTest in the location

       Create a service descriptor file with the MBean Class name, the name and type of service and copy this file into the above directory.
    c. If this service is dependent on other services, use the dependent element in the Service Descriptor:

       PubSubTestAdapterMBean0-service.xml:

```
<server>

      <mbean code="com.cunstomMBean.mq.fiorano.PubSubTestAdapterMBean"
name="Iris:ServiceType=PubSubTest,Name=PubSubTestService" xmbean-
dd="com/customMBean/mq/fiorano/PubSubTestAdapterMBean.xml">

          <depends
>Fiorano.mq.pubsub:ServiceType=PubSubManager,Name=TopicSubSystem</depends>

      </mbean>

          </server>
```

    d. If the service on which this is dependent is not known, configure this service to start after all other Server services have started. as stated in step 3

Sample 2:

```
<server>

   <mbean code="com.customMBean.mq.fiorano.PubSubTestAdapterMBean"
name="Iris:ServiceType=PubSubTest,Name=PubSubTestService" xmbean-dd="com/
customMBean /mq/fiorano/PubSubTestAdapterMBean.xml">

   </mbean>
```

```
</server>
```

3. **Deploy Service:** (See Sample 3.) [Link?]
    a. If the required dependencies have been specified in the service descriptor file then this file can be appended to any the FioranoMQ.lst file. (Note: The sorting order is defined by the dependency specified not by the location of the dependency in the file.)
    b. If required dependencies have not been specified start the service after all other Server services are started. Create a separate list file and add service entries to this file. Add this file to the end of the list file entries in the deploy.xml file.

Sample 3: (Deploy.xml)

```
<?xml version="1.0" encoding="UTF-8"?>

<list>FMQBootstrap.lst,FioranoMQ.lst,DefaultMQObjects.lst,TestService.lst</list>
```

4. **Start the FioranoMQ server**

Sample 4:

```
<mbean>

  <descriptors>

    <descriptor name="ComponentName" value="ComponentValue"/>

    …

    …

    …

  </descriptors>

  <class><!--Name of class --></class>

  <!--attributes-->

    <attribute access="read-only" getMethod="getStateString">

    <description>(no description)</description>

    <name>StateString</name>

    <type>java.lang.String</type>

    </attribute>

    …

    …
    ...

  <!--operations -->

    <description>Save the Current State</description>

    <name>save</name>

    <return-type>boolean</return-type>

    <descriptors>

    </descriptors>

    ...

    ...

    ...

</mbean>
```

30.2 Custom MBean Service Common Problems and Solutions

Common problems encountered while creating a custom MBean Service. See the solutions provided:

1. Problem:

ClassNotFoundException. Unable to find the MBean service.

Solution:

Set the custom service jar in the CLASSPATH either by editing fmq.bat or editing the system CLASSPATH.

2. Problem:

Service created was unable to register.

Solution:

Custom MBean must extend fiorano.jms.jmx.ComponentMBean.

3. Problem:

Start this as the last service of the Server (since this service is dependent on other services).

Solution:

Add this service in a separate list file and append this file to the end of the list files in the deploy.xml. See Sample 3.

4. Problem:

Custom code written in the lifecycle methods (create, start, and stop) is not getting called.

Solution:

Write the custom code in the createService(), startService(), stopService() instead of in the create, start and stop methods.

5. Problem:

Deployed the service in the service, but it is taking too long to perform a lookup for CF & Destinations.

Solution:

Set the property TransportProtocol to LPC for FioranoJNDIContext and execute the lookup. Without this change, the normal lookup is performed.

6. Problem:

XML metadata not found for the class

Solution:

The .xml file is not available at the correct location. Add the XML file in the same location as the class file. Create the jar again and copy it to the location mentioned in fmq.conf.

Chapter 31: Miscellaneous Features

31.1 Support for Destination Level Configuration

The parameters set for the destinations (Queues/Topics) in FioranoMQ can be configured in two ways:

- Subsystem Level Configuration
- Destination Level Configuration

By default, in any FioranoMQ profile, the parameters are fetched from the Subsystem level only. However, if some parameters are changed for a particular destination in the FioranoMQ profile at the destination level, then higher priority is given to this and this value is reflected on the destination instead of the value set at Subsystem level. To understand this clearly, please refer to the following example:

Consider the parameter InMemoryBufferSize that can be set for a Queue. The default value for this parameter is 1048576 and this parameter takes long values as input. If FioranoMQ Server is started with the default profile, then this default value gets assigned to all the queues.

If the value of this parameter is changed for the Queue PRIMARYQUEUE to, for example, 10485760, then the value of this parameter for this queue will be the changed value, but for all the other queues it will remain default value, 1048576.

Note: Any change in the values for any parameters in the Subsystem level will be reflected in all destinations. The same rule follows in the case of Topics and Topic Subsystem as well.

31.2 N Failover URL Support

Support for N failover URLs has been incorporated in FioranoMQ. This feature backup support of N other servers to be specified. If the primary server goes down, all the connected clients connect to one of the N failover servers. If this failover server goes down, then all the clients connect to one of the N-1 servers. If one/any of the servers that are down, come(s) up, then the number of options for the clients, in terms of servers that they can connect to, increases by one.

Usage

Failover URLs are specified while creating the Connection Factories. While creating connection factories through Fiorano Admin Studio, the failover URLs are set using the Backup Connect URLs parameter. Please refer to section 4.7.1 Creating a Connection Factory. While creating Connection Factories through the code, the failover URLs are set using the setSecondaryConnectURLs (String parameter) API in ConnectionFactory class.

The N FailoverURL support has been extended to the lookup operation as well. Client applications can set a semi-colon separated string of urls (syntax given below) in the InitialContext environment. If the primary server is unavailable for any reason, the FioranoMQ runtime looks up the same object from the server running on the backupURL.

Syntax

SecondaryURLs string is a list of valid URLs separated by semi colons.

Below is a valid entity:

http://schumacher:1856;http://barrichello:1956;http://ayrton:2056

Below is an invalid entity:

http://schumacher:1856:http://barrichello:1956:http://ayrton:2056

Sample Scenario

Consider a scenario where there are four FioranoMQ Servers.

(Server1, Server2, Server3, and Server4.)

1. server1 running on HTUhttp://schumacher:1856UT
2. server2 running on http://barrichello:1956
3. server3 running on Uhttp://ayrton: 2056
4. server4 running on HUhttp://ferrari:2156U

Create a QueueConnectionFactory (myqcf) on server1 with

- primaryURL = http:// schumacher:1856,
- secondaryURLs = http://barrichello:1956;http://ayrton:2056; http:// ferrari:2156,
- clientId=fiorano,
- description = a queue connection factory1.

Client1 (any Sender/Receiver) connects to server1 (looks up myqcf) and starts sending/receiving messages. Now, if server1 goes down, then client1 uses the revalidate method to connect to the next available running server (server2 in this case). Similarly, if server2 goes down, then client1 connects to server3, and so on. Clients can connect to any running servers that are specified in the list of primaryURL or SecondaryURL. If a server that had gone down comes up, it is rendered available for client connection once again.

Code Snippet

```
Hashtable env = new Hashtable();

env.put(Context.SECURITY_PRINCIPAL, "anonymous");

env.put(Context.SECURITY_CREDENTIALS, "anonymous");

env.put(Context.PROVIDER_URL, "http://164.164.131.1:1856;http://localhost:

1856");

env.put(Context.INITIAL_CONTEXT_FACTORY,

"fiorano.jms.runtime.naming.FioranoInitialContextFactory");
```

```
env.put("BackupConnectURLs",

"http://localhost:1956;http://localhost:2056;http://localhost:1656");

InitialContext ic = new InitialContext(env);

System.out.println("Created InitialContext :: " + ic);

// 1.1 Lookup Connection Factory and Topic names

TopicConnectionFactory topicConnectionFactory = (TopicConnectionFactory)

ic.lookup("multiURLTCF");

Topic topic = (Topic) ic.lookup("primaryTopic");
```

31.3 Advisory Message Listener

An Advisory Message Listener object is used to listen for advisory messages. Applications wishing to listen for advisory messages should register their listeners on the JMS connection object.

FioranoMQ provides advisory messages to inform its client applications about situations that may affect them. The advisory messages provide errors and informational messages.

Advisory Messages

Below are some of the advisory messages supported by FioranoMQ:

- **CONNECTION_STATE_CHANGED Event**: Indicates a change in the state of Connection object.
- **CSP_STATE_CHANGED**: Indicates that forwarding of messages stored in persistent storage on the client side is starting or has been completed.
- **CSP_ERROR_MESSAGE**: Indicates that an error has occurred while forwarding the stored messages.
- **CLIENT_APP_ERROR_MESSAGE**: Indicates that an unhandled exception has been detected by FioranoMQ's RTL.

CONNECTION_STATE_CHANGED Event

This event is raised when the state of the connection object is changed. Below are the possible states of a connection object:

- **FMQConnectionState.CONNECTION_ACTIVE:** This state indicates that the connection is alive and available.
- **FMQConnectionState.CONNECTION_STARTED:** This state indicates that the connections are active and ready to deliver messages to the callback listeners. ACTIVE and STARTED are the same with the only difference reflected in consuming messages. For example, connection start() is invoked for consuming the messages and thus a different state is introduced.

- **FMQConnectionState.CONNECTION_REVALIDATING:** This state indicates that the connection is down, but indicates that the fmq client library is attempting to reconnect to the server.
- **FMQConnectionState.CONNECTION_DISCONNECTED:** This state indicates that the connection is down and the fmq client library has failed to reconnect to the server in the pre-specified attempts.
- **FMQConnectionState.CONNECTION_CLOSED**: This state indicates that the connection is in a closed state.

CONNECTION_STATE_CHANGED Event is raised for a durable connection. For a non- durable failed connection, the fmq client library throws a JMSException in the APIs (requiring a call to the Server). Client applications listening for advisory messages can handle this event as per their application programming. One of the possible methods of handling such a situation is to log this event.

Below is the sample code that prints the connection state on the console:

Step 1: When an advisory message is received it checks whether the event is a Connection state change event. Type cast the message into the required FMQConnectionState object on verification.

if (advisoryMessage.getType () == advisoryMessage.CONNECTION_STATE_CHANGED)

FMQConnectionState connStateChange = (FMQConnectionState) advisoryMessage;

Step 2: Using the above FMQConnectionState object, the current state of the connection object is printed. Possible values that maybe returned are:

FMQConnectionState.CONNECTION_ACTIVE, FMQConnectionState.CONNECTION_STARTED , FMQConnectionState.CONNECTION_REVALIDATING, FMQConnection-State.CONNECTION_DISCONNECTED, FMQConnectionState.CONNECTION_CLOSED

System.out.println ("Current state ->“+ connStateChange.getState ());

Step 3: Prints whether the connection is active. This API compares getState () with FMQConnectionState.CONNECTION_ACTIVE or FMQConnectionState.CONNECTION_STARTED.

System.out.println ("Active ->“+ connStateChange.isActive ());

Step 4: Prints whether the connection has been disconnected. This API compares getState () with FMQConnectionState.CONNECTION_DISCONNECTED.

System.out.println ("Disconnected ->“+ connStateChange.isDisconnected ());

Step 5: Prints whether the connection attempts to reconnect to the server. This API compares getState () with FMQConnection-State.CONNECTION_REVALIDATING.

System.out.println ("Is Revalidating -> " + connStateChange.isRevalidating ());

Step 6: Prints the URL to which the client is connected. If the client is not connected to a server ‘null’ is printed.

System.out.println ("Connect url ->“+ m_connection.getConnectUrl ());

Step 7: Prints the list of backup URLs used in revalidation of the connection.

String [] backupUrls = m_connection.getBackupUrls ();

if (backupUrls! = null)

{

for (int i = 0; i < backupUrls.length; i++)

System.out.println ("Backup Url ->“+ backupUrls[i]) ;}

CSP_STATE_CHANGED

This message indicates a change in the state of the persistent store at the client end.

CSP_STATE_Changed is raised in the situations listed below:

- When the FioranoMQ client library has started transferring messages stored in the persistent store at the client side.
- When the FioranoMQ client library has transferred all the messages stored in the persistent store at the client end.

CSP_STATE_CHANGED is raised only for a durable connection. For non-durable connections, messages are not stored in the store at the client side.

Client applications listening for advisory messages can handle this event as per their application programming. One possible way of handling this situation is to log this event. is a sample of the code that prints the CSP_STATE_CHANGED on the console:

Step 1: When an advisory message is received, the code checks whether it is a CSP_STATE_CHANGED message. ype casts the message into the required CSP_STATE_CHANGED object on verification.

if (advisoryMessage.getType () == advisoryMessage.CSP_STATE_CHANGED)

FioranoCSP_STATE_CHANGED CSP_STATE_CHANGED =

(FioranoCSP_STATE_CHANGED) advisoryMessage;

Step 2: Prints whether the fmq client library has started transferring the stored messages. If messages are currently being transferred, this API returns **true**. If messages are not being transferred it returns **false**.

System.out.println ("Is transferring -> " + CSP_STATE_CHANGED.isTransferring ());

Step 3: Prints whether the fmq client library has transferred all the locally stored messages. It returns **true** if the transfer is complete and **false** if the transfer is not complete.

System.out.println ("Transfer complete ->“+

CSP_STATE_CHANGED.isTransferComplete ());

CSP_ERROR_MESSAGE

This message indicates that an unexpected error occurred while forwarding the messages stored in the persistent store at the client end. Client applications log the error message so that the application administrators can investigate the cause of the error.

Below is a sample of the code that prints the CSP_ERROR_MESSAGE on the console.

Step 1: When an advisory message is received, the code checks whether it is a CSP_ERROR_MESSAGE or not. Type casts the message into the required CSP_ERROR_MESSAGE object on verification.

if (advisoryMessage.getType () == advisoryMessage.CSP_ERROR_MESSAGE)

FioranoCSPErrorEvent CSP_ERROR_MESSAGE = (FioranoCSPErrorEvent)

advisoryMessage;

Step 2: Prints the exception stack on the console.

if (CSP_ERROR_MESSAGE.getException ()! = null)

CSP_ERROR_MESSAGE.getException ().printStackTrace ();

Step 3: Gets the message for csp corruption

CSP_ERROR_MESSAGE.getErrMessage(); CLIENT_APP_ERROR_MESSAGE

This message indicates that an unhandled exception has been detected by FioranoMQ's RTL. An exception is thrown in the onMessage() method (a method in MessageListener of the MessageConsumer) of the user's code if the code has not been handled properly. This is actually a programming error in the client application. If such an uncaught exception inside the onMessage() method of the client application's MessageListener code occurs, it raises an advisory message Client_App_Error_Message. A MessageListener can be registered with a MessageConsumer for the purpose of receiving messages asynchronously.

The earlier versions of FioranoMQ's RTL handled this situation differently. When an exception inside the onMessage() method of the client code occured, the connection's ExceptionListener was notified through the invocation of its onException() method.

The sample code,below, demonstrates the registrationi of an advisory message listener:

Step 1: Create a .jms connection.

Connection connection = connectionFactory.createConnection ();

Step 2: Create an advisory message listener.

MyAdvisoryMessageListener listener = new MyAdvisoryMessageListener ();

Step 3: Set the advisory message listener on the connection object.

((fiorano.jms.runtime.IFMQConnection) connection).setAdvisoryMessageListener (listener);

For more details on Advisory Message APIs, please refer to the javadocs of the fiorano.jms.runtime package.

31.4 Message Browser Support in FioranoMQ

FioranoMQ provides support for message browsing by a durable subscriber registered with the FioranoMQ Server on any topic. A MessageBrowser on a TopicSession using the ClientID and SubscriberID of the concerned subscriber as the arguments can be created.

Once this MessageBrowser is created, it can be used to retrieve all messages that are stored in the FioranoMQ Server for that particular DurableSubscriber. The browser facility works when the concerned Durable Subscriber is passive.

Browser APIs

FioranoMQ provides public APIs for browsing messages on a durable subscriber. The sample code, below, describes how these APIs are used:

For creating a MessageBrowser:

```
FioranoMessageBrowser msgBrowser =

((FioranoTopicSession)topicSession).createMessageBrowser(clientID,subID);
```

For enumerating the messages for this browser:

```
Enumeration enum = msgBrowser.getEnumeration();

while(enum.hasMoreElements())

Message msg = (Message)enum.nextElement();
```

31.5 FioranoMQ - XML Interoperability

The XMLInteroperability Toolkit, which is a part of FioranoMQ, allows the interoperability of FioranoMQ across JMS Vendors. The Toolkit provides APIs with tools required to convert a JMS Message to an XML document and to recreate a Message from an XML document. The toolkit currently supports XML conversions for TextMessage and MapMessage. The toolkit is generic and supports XML Parser plug-ins at runtime. This toolkit can be used to communicate across different JMS Vendors. The DTD (Document Type Definition) of the Messages is public and can be used by any non-Fiorano JMS vendor to communicate with FioranoMQ.

The contents of this chapter are organized as listed below:

- JMS Message to XML.
- XML to JMS Message.
- Using XMLAdapter Toolkit of FioranoMQ with other JMS Vendors.

31.5.1 JMS Message to XML

A JMS TextMessage or MapMessage can be converted to XML. To convert a message to XML, an appropriate XMLAdapter has to be created. If the Message to be converted is a MapMessage, then an instance of XMLMapMessage-Adapter is created. To illustrate, pick a TextMessage and convert it to an XML Document. Create an instance of the XMLTextMessageAdapter:

```
XMLTextMessageAdapter adapter = new XMLTextMessageAdapter();
```

To represent this message as an XML, invoke the following method on the adapter:

```
StringBuffer xmlData = adapter.createXML (txtMsg, false);
```

This call results in the conversion of the Message into an XMLDocument based on the DTD for text messages.

The code snippet used is:

```
XMLTextMessageAdapter adapter

= new XMLTextMessageAdapter();

TextMessage textmsg2 = (TextMessage)msg;

// Create an XML of the received Text Message and indicate

// that the DTD should be embedded into the XML

StringBuffer xmlData = adapter.createXML

(textmsg2, true);

// Dump the Document to a file

RandomAccessFile rf = new RandomAccessFile

(“message.xml” ,”rw”);

rf.writeUTF(xmlData.toString());

rf.close ();
```

31.5.2 XML to JMS Message

An XML Document representing a JMSMessage can be reconverted back to a JMSMessage. For example, consider an XML Document representing a TextMessage. Create an instance of the XMLTextMessageAdapter:

```
XMLTextMessageAdapter adapter

= new XMLTextMessageAdapter ();
```

Associate the XML Parser to the adapter. By default IBM's XML Parser is used. Specify the name of the main class of the parser, represented as an argument to this method. The Toolkit uses Java's reflection methods to create an instance of the parser

adapter.setXMLParser

("org.apache.xerces.parsers.SAXParser");

Reconvert the XML to a message:

adapter.createMessage(textmsg, "specify the URI of the XML Document")

The code snippet used is:

// Create the Adapter

XMLTextMessageAdapter adapter

= new XMLTextMessageAdapter ();

// Set the parset to be used

adapter.setXMLParser

("org.apache.xerces.parsers.SAXParser");

// Create a text message for used to store the XML

TextMessage textmsg1 = session.createTextMessage();

// Convert the message back to XML

m_adapter.createMessage (textmsg1, "URI of the XML Doc");

31.5.3 Using XMLAdapter Toolkit of FioranoMQ with other JMS Vendors

By default, the XML Interoperability toolkit is configured to work with FioranoMQ. This is possible because the toolkit allows a **provider specific adapter** to be set to the XMLAdapter. Any other JMS Vendor can use this toolkit to expose JMS messages by implementing the IProviderSpecificMessageAdapter interface and plugging it to the XMLAdapter. (More samples can be found in the \fmq\samples\XMLToolkit folder of the FMQ installation directory.)

class XMLTextMessageAdapter Declaration

public XMLTextMessageAdapter()

Purpose

The XMLTextMessageAdapter provides a compete set of methods to convert a Text-Message to its corresponding XML document and vice versa.

Method Summary

public XMLTextMessageAdapter()

Constructor: Creates an XMLTextMessageAdapter.

public void setXMLParser (String xmlParser)

throws JMSException.

Sets the fully qualified name of the XML Parser, to be used to parse the JMS message.

public void setProviderSpecificAdapter

(IProviderSpecificMessageAdapter adapter)

throws JMSException.

Sets a provider specific adapter. By default the XML toolkit works with the adapter of FioranoMQ. Any other JMS Implementation can interoperate by setting and implementing the appropriate adapters.

public void createMessage

(TextMessage msg, String uri)

throws JMSException

public void createMessage

(TextMessage msg, InputSource source)

throws JMSException

Converts the XML Message, identified by the InputSource, or a String representation, into a JMS text message.

public StringBuffer getMessageBody

(TextMessage msg)

throws JMSException

Returns the header as an XML document from the JMS message.

public String getMessageBodyDTD()

Returns the DTD for the message body.

31.6 XMLMapMessageAdapter

Declaration

public XMLMapMessageAdapter()

Purpose

The XMLMapMessageAdapter provides a complete set of methods to convert a Map-Message to its corresponding XML document and vice-versa.

Method Summary

public XMLMapMessageAdapter()

Constructor: Creates an XMLMapMessageAdapter.

public void setXMLParser (String xmlParser)

throws JMSException.

Sets the fully qualified name of the XML Parser to be used to parse the JMSMessage.

public void setProviderSpecificAdapter (IProviderSpecificMessageAdapter adapter)

throws JMSException.

Sets a provider specific adapter. By default the XML toolkit works with the adapters of FioranoMQ. Any other JMS Implementation can interoperate by setting and implementing the appropriate adapters.

public StringBuffer createXML(MapMessage msg,

boolean embedded) throws JMSException.

Converts the Message object to its XML representation. Raises a JMSException if the conversion fails.

public void createMessage(MapMessage msg, String uri)

throws JMSException

public void createMessage(MapMessage msg,

InputSource source) Throws JMSException

Converts the XML Message identified by the InputSource or a String representation of the URL into a JMSMessage.

public StringBuffer getMessageBody(MapMessage msg)

throws JMSException

Returns the header as an XML document from the JMS message.

public String getMessageBodyDTD()

Returns the DTD for the message body.

31.7 IProviderSpecificMessageAdapter

Purpose

The IProviderSpecificMessageAdapter defines the signature of all the methods that are defined by the provider's adapters necessary for useing FioranoMQ XML Adapters. JMS does NOT provide methods to retrieve properties that are stored as part of the message. This interface extends the JMSMessage behavior providing methods to retrieve the 'types' of properties stored.A complete set of methods is furnished to convert a MapMessage to its corresponding XML document and vice versa.

Other JMS Vendors can make use of the XML Toolkit by implementing this Adapter.

Method Summary

public byte getMessagePropertyType(Message message,

String propertyKey) throws JMSException

Retrieves the MessageProperty Type for the specified propertyKey.

public byte getMapMessagePropertyType (MapMessage message,

String propertyKey) throws JMSException

Retrieves the **Property Type** of the Map message property

31.8 Integration with Spring Framework

The Spring Framework is an open source application framework for the Java platform. The core features of the Spring Framework can be used by any Java application, but there are extensions for building web applications on the Java Enterprise platform. Although the Spring Framework does not impose any specific programming model, it has become popular in the Java community as an alternative to, replacement for, or an addition to the Enterprise JavaBean (EJB) model.

This section explains how to integrate FioranoMQ client samples with Spring Framework.

The resources and libraries listed below are needed:

- FioranoMQ Server
- ant
- Jars --

Please download the following jars and store them in a directory named "lib" in the current sample's home folder.

- spring.jar (http://www.java2s.com/Code/Jar/Spring-Related/spring.jar.htm).
- commons-logging.jar (http://www.java2s.com/Code/Jar/jboss-5.0.0.Beta2/commons-logging.jar.htm).

The jars listed below are shipped along with FioranoMQ: (Modify the build files to include these jars only when the sample is moved from its default location.)

- fmq-client.jar
- fiorano-framework.jar
- jms.jar

The Samples for the Spring Framework are present in the directories listed below:

<FMQ_HOME>/samples/PTP/SpringSamples

<FMQ_HOME>/samples/PubSub/SpringSamples

The sample directory comprises of the two directories listed below:

- The "lib" directory should contain the necessary jars.
- The "src" directory contains the java samples and the Spring configuration files.

The Spring configurations are present under the directory "src/spring/".

These xml files provide the samples with configurations like destination, ContextFactory, ConnectionFactory and so on.

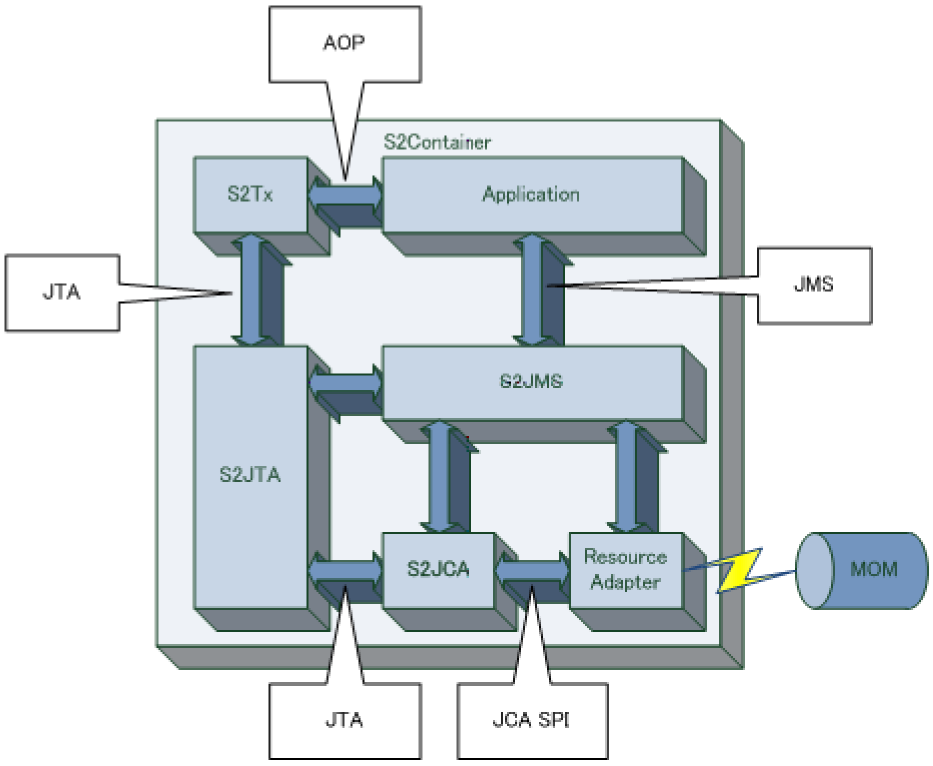
The java samples are present under the directory "src/springexample/client". These java files pickup configurations from xml files.

Samples can be run using the following command from the sample's home directory: **ant -f build-jmsreceiver.xml**. A detailed explanation can be found in the readme files provided in the samples directories.

31.9 Integration with Seasar Framework

Seasar 2, the most popular Dependency Injection (DI) open source framework in Japan. S2Container (Sesar2) provides set of components called S2JMS (Sesar2JMS) to easily build applications to send and receive messages through JMS.

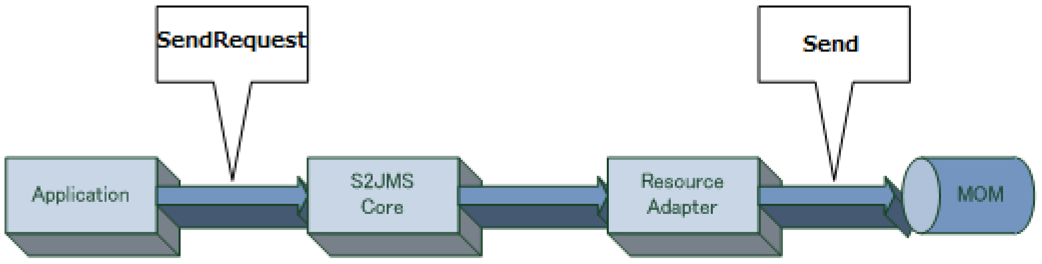
A Sesar container supports Distributed transactions (JTA) and JMS connection pooling. Sesar connects to MOM vendors through Resource Adapter.

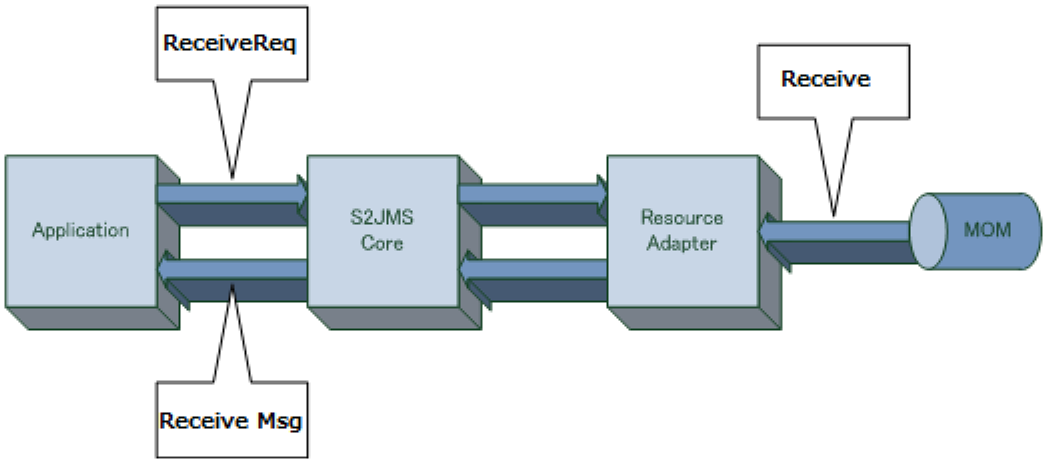


31.9.1 Outbound Communication

Outbound Communication is a form of sending and receiving messages from MOM via resource adapter.

In Outbound communication, Application creates the connection to MOM with API javax.jms.ConnectionFactory # getConnection () before sending or receiving messages.

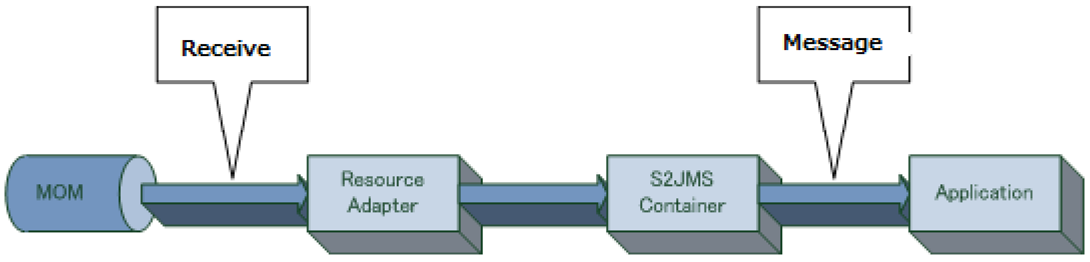




31.9.2 Inbound Communication

Inbound Communication is a form of receiving messages asynchronously from MOM via resource adapter.

In Inbound communication, resource adapter creates the connection to MOM. When Messages is available in MOM, Resource adapter notifies the application with Message.



This section explains how to integrate FioranoMQ client samples with Sesar Framework. Fiorano Sesar samples are created as Eclipse java project so can be imported to Eclipse IDE.

Fiorano Sesar samples are developed with Seasar S2JMS-1.0.1 version. Please refer to http://www.seasar.org/index.html for latest version

The resources and libraries listed below are needed:

- FioranoMQ Server
- Fiorano Resource Adapter
- fmq-client.jar
- fiorano-framework.jar
- jms.jar

The Samples for the Seasar Framework are present in the directories listed below:

<FMQ_HOME>/samples/SeasarJMSSample

The sample directory comprises of the two directories listed below:

- The "lib" directory should contain the necessary jars.
- The "src" directory contains the java samples and the Spring configuration files.

The above listed jars are shipped along with FioranoMQ. Copy fmq-client.jar, fiorano-framework.jar, jms.jar to **lib** directory.

Generate Fiorano Resource Adapter (fmq-connector-ra.rar) and copy the same to **lib** directory. Refer to “Deployment of FioranoMQ Resource Adapter” section for more detail about Resource Adapter.

The Seasar configurations are present under the directory "src".

.dicon files provide the samples with configurations like destination, ContextFactory, ConnectionFactory, XATrasanctions and so on. Detailed explanation can be found as inline comments in .dicon files.

The java samples are present under the directory "src/seasar". These java files pickup configurations from .dicon files.

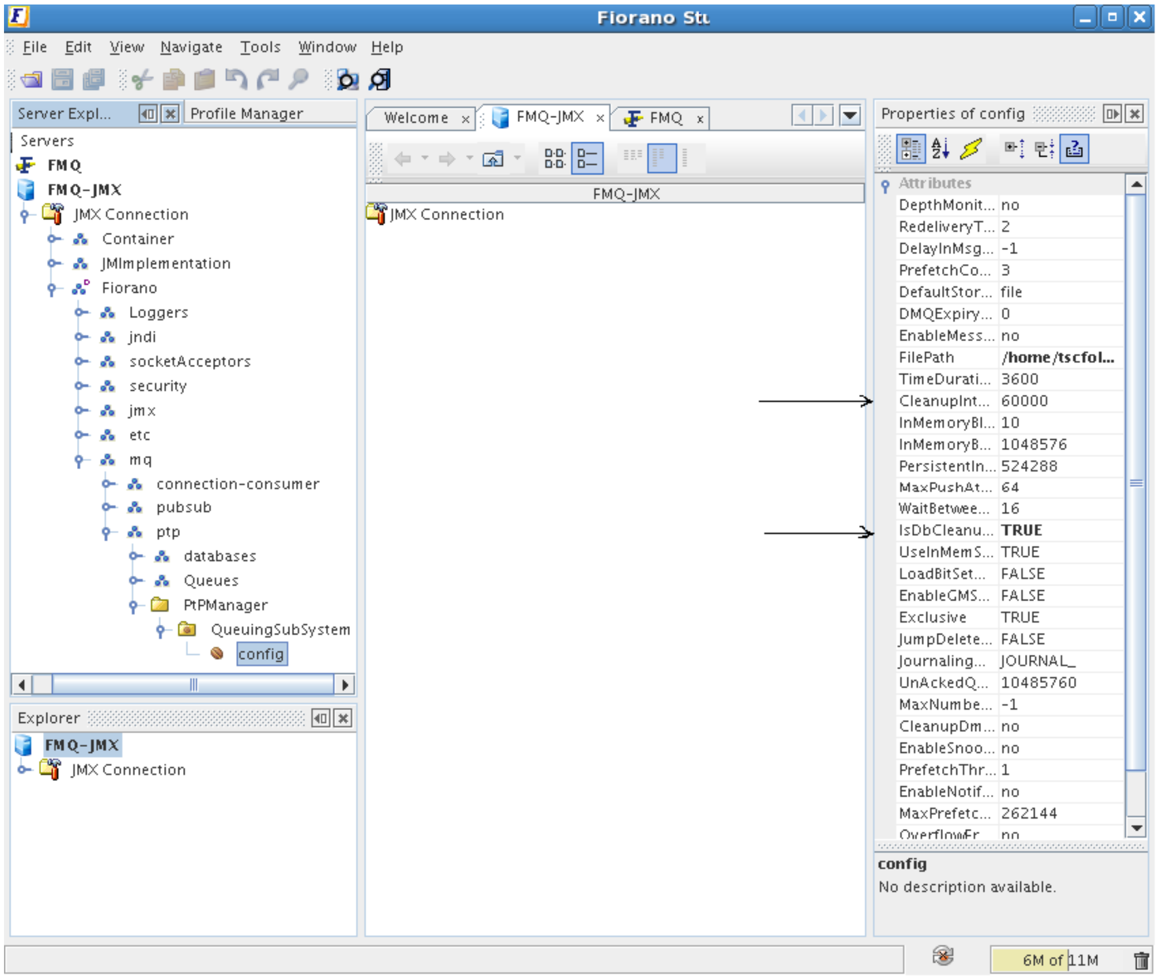
Samples can be run and debugged from Eclipse IDE.

31.10 Message Expiry

A message may get expired based on its Time To Live (TTL) value. For more information on how a message expires, refer **Chapter 8: Message Expiry** in *Fiorano MQ Concepts Guide*.

31.10.1 Purging Expired Messages in Queues

Expired messages in the queues can be cleaned up by setting the flag 'IsDbCleanupEnabled' to 'True', by default this is set to 'false'. To configure this parameter, login through FMQ-JMX in Fiorano Studio. Navigate to JMXConnection--> Fiorano--> mq--> ptp --> PTPManager --> QueueSubsystem --> config. The parameter can be changed here. Refer to the screenshot shown below. The changes need to be saved and server should be restarted. The frequency with which the server checks for expired messages is again configurable through the parameter CleanupInterval (defaults to 10 minutes). You can configure this value at Subsystem level or at destination levels.



DepthMonit...	no
RedeliveryT...	2
DelayInMsg...	-1
PrefetchCo...	3
DefaultStor...	file
DMQExpiry...	0
EnableMess...	no
FilePath	/home/tscfol...
TimeDurati...	3600
CleanupInt...	60000
InMemoryBl...	10
InMemoryB...	1048576
PersistentIn...	524288
MaxPushAt...	64
WaitBetwee...	16
IsDbCleanu...	**TRUE**
UseInMemS...	TRUE
LoadBitSet...	FALSE
EnableGMS...	FALSE
Exclusive	TRUE
JumpDelete...	FALSE
Journaling...	JOURNAL_
UnAckedQ...	10485760
MaxNumbe...	-1
CleanupDm...	no
EnableSnoo...	no
PrefetchThr...	1
EnableNotif...	no
MaxPrefetc...	262144
OverflowFr...	no

config
No description available.

Note: The cleanup handler is not present for Topics. This is only available for Queues. For now, this has to be done manually by running a browser on to a topic which has expired messages piled up.

If a browser is made to run on a topic, all the expired messages will get cleaned up. You will need to write a Topic browser application which runs on a specific topic in a regular interval of time. Refer to the Message Browser sample located at $Fiorano_Home/fmq/samples/PubSub/MessageBrowser for writing a browser sample on Topics. Purging of the expired messages happens only when the server is in **active state.**

Copies of the expired messages in any of the destinations can be stored in a queue named **SYSTEM_DEADMESSAGES_QUEUE**. For more information on this queue, refer to **Chapter 13: Dead Message Queu**e.

31.11 Poison Message Handling

31.11.1 Poison Messages

Sometimes, a badly-formatted message arrives on a destination. In this context, badly-formatted means the receiving application cannot process the message correctly. Such message can cause the receiving application to fail and to back out this badly-formatted message. The message can then be repeatedly delivered to the input queue and repeatedly backed out by the application. These messages are known as **Poison Messages**.

Once the message re-delivery attempts becomes greater than the maximum re-delivery attempts (maxRedeliveryTriesOnListenerException), then the MessageConsumer/ConnectionConsumer detects that it is a poison message and reroutes it to an alternative configurable destination.

Poisonous messages will be re-routed to an alternative destination only for receivers/subscribers created from Session with AUTO_ACKNOWLEDGE /DUPS_OK_ACKNOWLEDGE acknowledgment mode.

AllowDurableConnection and enableAutoRevalidation flags at the server level must be set to true in order to send the poisonous message to the FioranoMQ server even if the connection is disconnected.

AllowOnTheFlyCreationOfDestinations must be set to true in order to send thepoison message to the configured destination."

Note: If a non-durable subscriber receives persistent/non-persistent messages and durable subscriber receives non-persistent messages and if that message is determined as poisonous message, it will be dropped if the connection with FioranoMQ server is disconnected while sending it.

31.11.2 Configurable Parameters at Queue Subsystem Level

a) enablePoisonMessageMonitoring:

This parameter must be set to true in order to enable poisonous message handling.

- type : boolean
- Default value : false
- Restart required : yes

b) poisonMessageDestination:

This parameter is used for specifying the PTP-destination for routing the poisonous messages.

- type : String
- Default value : SYSTEM_BACKOUT_QUEUE
- Restart required : yes

31.11.3 Logging

In order to log information while sending the poisonous message, the logger named "log4j.logger.Fiorano.FMQ.Services.ClientRootLoggerServices.FMQClientLoggerServices.PTP.PoisonMessageHandler" must be enabled in log4j.properties which can be located under $FIORANO_HOME/fmq/bin.

C, C++,C# client runtime library also supports poison message handling.

For more information on FioranoMQ Client Logging, please refer section 21.3 Fiorano Client Logger and 28.4 FioranoMQ Client Logging in FioranoMQ Handbook.

31.12 Shared Subscriptions

A shared non-durable subscription is used by a client which enables to share the work of receiving messages from a durable subscription amongst multiple consumers. A shared durable subscription can therefore have more than one consumer. Each message from the subscription will be delivered to only one of the consumers on that subscription.

A shared durable subscription is created, and a consumer created on that subscription, as specified by the JMS specification. A shared durable subscription is identified by a name specified by the client and by the client identifier if set. If the client identifier is not set for a JMS connection, by default, Fiorano assigns a provider specific identifier such as a string starting with "JMS_FIORANO_CLIENT_ID_" followed by a numerical value that is unique for each such connection.

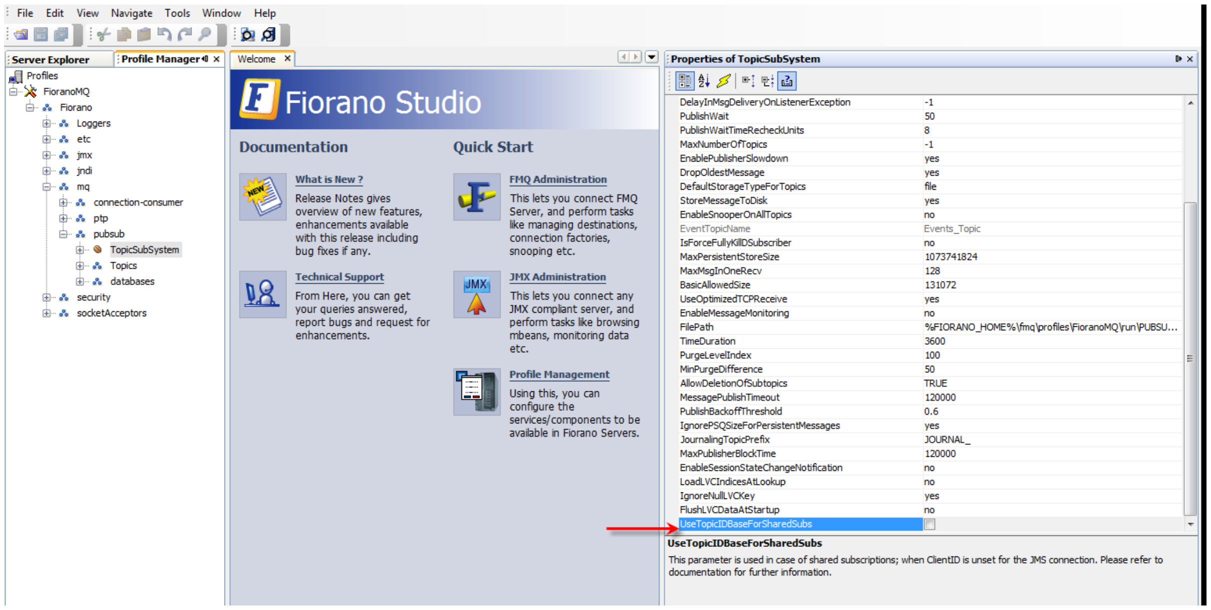
In case of shared durable subscriptions, it is not essential to set client identifier for the JMS connection. So, if client applications are coded in such a way and as Fiorano provider assigns a default identifier which changes randomly for each instance of connection creation, there will be a NEW durable subscription created each time an application goes offline and comes up which is erroneous.

In order to resolve this issue, in order to identify a shared durable subscription Fiorano provider assigns a secondary identifier (to a JMS connection for which client identifier is unset). This name can be unique for all shared durable subscriptions across the JMS provider or unique for all subscriptions under a topic. This behavior can be changed by the parameter which can be configured in the following way -

31.12.1 Online Configuration

- Open Admin Studio
- Login through FMQ-JMX and traverse to Fiorano > mq > pubsub > PubSubManager > TopicSubSystem > config.
- Modify 'UseTopicIDBaseForSharedSubs' parameter
- Save configuration and restart server

31.12.2 Offline Configuration



Release Notes gives overview of new features, enhancements available with this release including bug fixes if any.

This lets you connect FMQ Server, and perform tasks like managing destinations, connection factories, snooping etc.

From Here, you can get your queries answered, report bugs and request for enhancements.

This lets you connect any JMX compliant server, and perform tasks like browsing mbeans, monitoring data etc.

Using this, you can configure the services/components to be available in Fiorano Servers.

Properties of TopicSubSystem

DelayInMsgDeliveryOnListenerException	-1
PublishWait	50
PublishWaitTimeRecheckUnits	8
MaxNumberOfTopics	-1
EnablePublisherSlowdown	yes
DropOldestMessage	yes
DefaultStorageTypeForTopics	file
StoreMessageToDisk	yes
EnableSnooperOnAllTopics	no
EventTopicName	Events_Topic
IsForceFullyKillDSubscriber	no
MaxPersistentStoreSize	1073741824
MaxMsgInOneRecv	128
BasicAllowedSize	131072
UseOptimizedTCPReceive	yes
EnableMessageMonitoring	no
FilePath	%FIORANO_HOME%\fmq\profiles\FioranoMQ\run\PUBSU...
TimeDuration	3600
PurgeLevelIndex	100
MinPurgeDifference	50
AllowDeletionOfSubtopics	TRUE
MessagePublishTimeout	120000
PublishBackoffThreshold	0.6
IgnorePSQSizeForPersistentMessages	yes
JournalingTopicPrefix	JOURNAL_
MaxPublisherBlockTime	120000
EnableSessionStateChangeNotification	no
LoadLVCIndicesAtLookup	no
IgnoreNullLVCKey	yes
FlushLVCDataAtStartup	no
UseTopicIDBaseForSharedSubs	☐

UseTopicIDBaseForSharedSubs

This parameter is used in case of shared subscriptions; when ClientID is unset for the JMS connection. Please refer to documentation for further information.

If this 'boolean' parameter is set to true (default value - false), the secondary client identifier will have the topic's name within it. In such a case, rather than using the normal way to unsubscribe a durable subscription from client application using only the subscription name, it must be done in the following way -

Java

```
TopicConnection topicConnection =
                         topicConnectionFactory.createTopicConnection();
TopicSession topicSession = topicConnection.createTopicSession(false, 1);
MessageConsumer tsub = topicSession.createSharedDurableConsumer(topic,

                         "Sample_Durable_Subscriber");
..
topicSession.unsubscribe(topicName, "Sample_Durable_Subscriber"); //changed
```

CPP

```
m_tc = m_tcf->createTopicConnection(m_usrName, m_usrPasswd);
m_ts = m_tc->createTopicSession(FALSE, AUTO_ACKNOWLEDGE);
m_subscriber = m_ts->createSharedDurableConsumer(m_topic,

                         "Sample_Durable_Subscriber");
..
m_ts->unsubscribe(topicName, "Sample_Durable_Subscriber"); //changed
```

31.13 JMSXDeliveryCount

When a client receives a message the mandatory JMS-defined message property JMSXDeliveryCount will be set to the number of times the message has been delivered.

How to access:

in javax.jms.Message:

```
int getIntProperty("JMSXDeliveryCount")
```

The purpose of the JMSXDeliveryCount property is to allow consuming applications to identify whether a particular message is being repeatedly redelivered and take appropriate action.

Note:

- Not applicable to Journalling and snooping.
- Not applicable to LVC.

JMSXDeliveryCount property is not persisted, so not guaranteed to be exactly correct in server restart/failover. This property is also not propogated to backup server for high availability deployments. In case of server failure, if message is redelivered, it will be delivered with JMSXDeliveryCount as 2.

Following cases on client side are considered are redelivery attempts:

1. MessageListener#onMessage() throwing RuntimeException: message will be redelivered with increased delivery count.
2. in trasacted session and CLIENT_ACK mode: session.rollback() and queueReceiver.recover() will cause message to redelivered with increased delivery count.

Note:

Above cases are considered to be client-side redeliveries and server is not notified about it for performance reasons.

31.14 Sending Messages Asynchronously

A Producer can send a message either synchronously or asynchronously. A normal synchronous send involves sending the message to FioranoMQ server and then waiting for an acknowledgement to be received before returning. An asynchronous send involves sending the message to FioranoMQ server and return without waiting for an acknowledgement. When the acknowledgement is received, FioranoMQ client runtime will notify the client application by invoking the onCompletion method on the application specified CompletionListener object. If for some reason the acknowledgement is not received, FioranoMQ client runtime will notify the application by invoking the CompletionListener's onException method.

Methods used for sending messages asynchronously:

Following methods can be used for sending messages asynchronously.

1. Under javax.jms.MessageProducer

a. send(Message message, CompletionListener completionListener)

b. send(Message message, int deliveryMode, int priority, long timeToLive, CompletionListener completionListener)

c. send(Destination destination, Message message, CompletionListener completionListener)

d. send(Destination destination, Message message, int deliveryMode, int priority, long timeToLive, CompletionListener completionListener)

2. Under javax.jms.JMSProducer

A JMSProducer may be used to send a message asynchronously by calling the method setAsync(CompletionListener completionListener) on the JMSProducer prior to calling one of the send methods.

Configurable parameters at Connection Factory level :

1. **AsyncSendBatchBufferSize** : Maximum amount of data (in bytes) that a JMS Session would store in un-acknowledged state for messages sent asynchronously. Attempt to exceed this size will result in either exception or publisher block as configured in PublisherBehaviourOnAsyncSendBufferOverflow parameter.
2. **AsyncSendCompletionWaitTimeout** : Maximum amount of time (in milli seconds) that a JMS Session would wait for acknowledgement for each message sent asynchronoulsy. CompletionListener 's onException will be invoked if no acknowledgement is received within this configured time.
3. **PublisherBehaviourOnAsyncSendBufferOverflow** : This parameter defines the behaviour of publisher when AsyncSendBufferSize is exceeded. A new publish call can either throw an exception or block.

Note :

1) If a message is sent to CSP it is considered as successful send and CompletionListener's onCompletion will be invoked. In case of any exception while storing it in CSP, CompletionListener's onException will be invoked.

2) If AllowDurableConnection is disabled and EnableAutoRevalidation is enabled and if connection with the server goes down with messages waiting for acknowledgement, messages will either be dropped or CompletionListener's onException will be invoked as configured in PublishBehaviourInAutoRevalidation parameter in connection factory. If PublishBehaviourInAutoRevalidation is set to "Block" then any new send call will be blocked till the connection revalidation is successful.

Chapter 32: FioranoMQ Web Management Tool

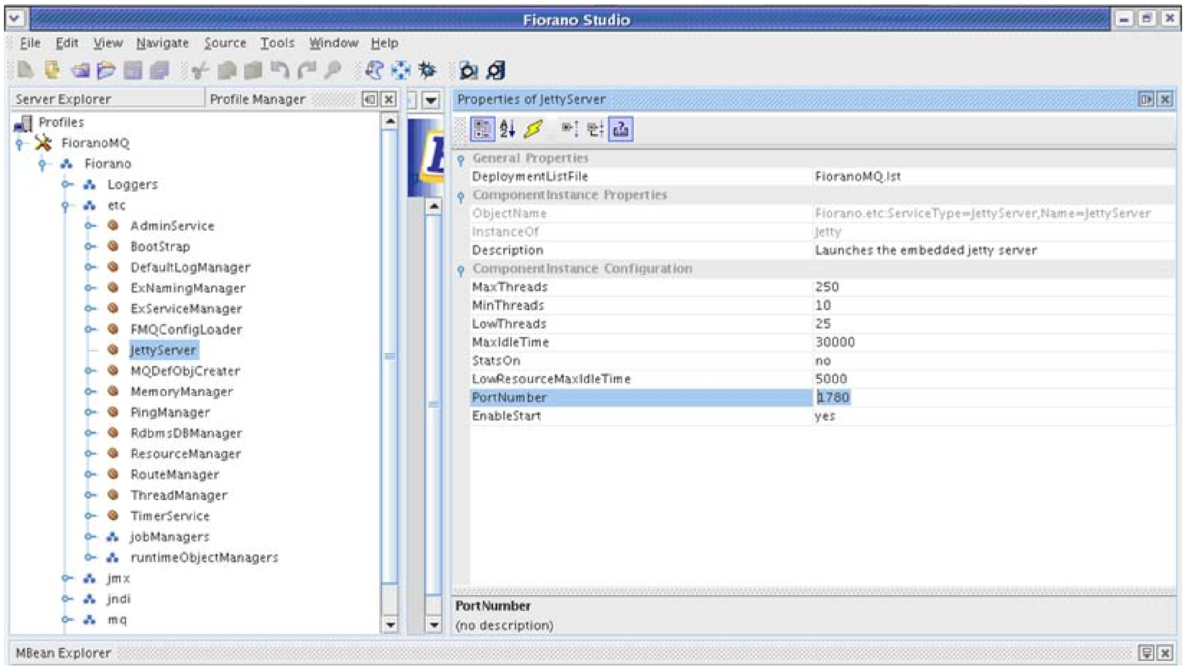
32.1 What is Web Management Tool

FioranoMQ Web Management (WMT) is a web based tool used to monitor and configure FioranoMQ Server in the online mode. The WMT connects various users to the FioranoMQ Server through a web browser allowing them to perform various operations. Toconnect to the FioranoMQ Server using this tool the JettyServer needs to be enabled in the FioranoMQ Server profile.

32.2 Configuring Web Manager

Follow the steps below to configure the JettyServer within FioraoMQ Server profile:

1. Open **FioranoMQ** profile in Studio. Navigate to **FioranoMQ-->Fiorano-->etc-->JettyServer**. You can see various properties in the **Properties of JettyServer** pane.



General Properties	
DeploymentListFile	FioranoMQ.lst
ComponentInstance Properties	
ObjectName	Fiorano.etc:ServiceType=JettyServer,Name=JettyServer
InstanceOf	Jetty
Description	Launches the embedded jetty server
ComponentInstance Configuration	
MaxThreads	250
MinThreads	10
LowThreads	25
MaxIdleTime	30000
StatsOn	no
LowResourceMaxIdleTime	5000
PortNumber	1780
EnableStart	yes

PortNumber
(no description)

Figure 32.1: Properties of JettyServer Pane

2. In **ComponentInstance Configuration,**

   - Set the **EnableStart** property to **yes** for using the Web Management tool. By default, this property is set to **yes**.
   - The default **PortNumber** is **1780**. All Web Management tool users connect to this port. This port number must be unique for every StandAlone FioranoMQ Server running on the same IP address.

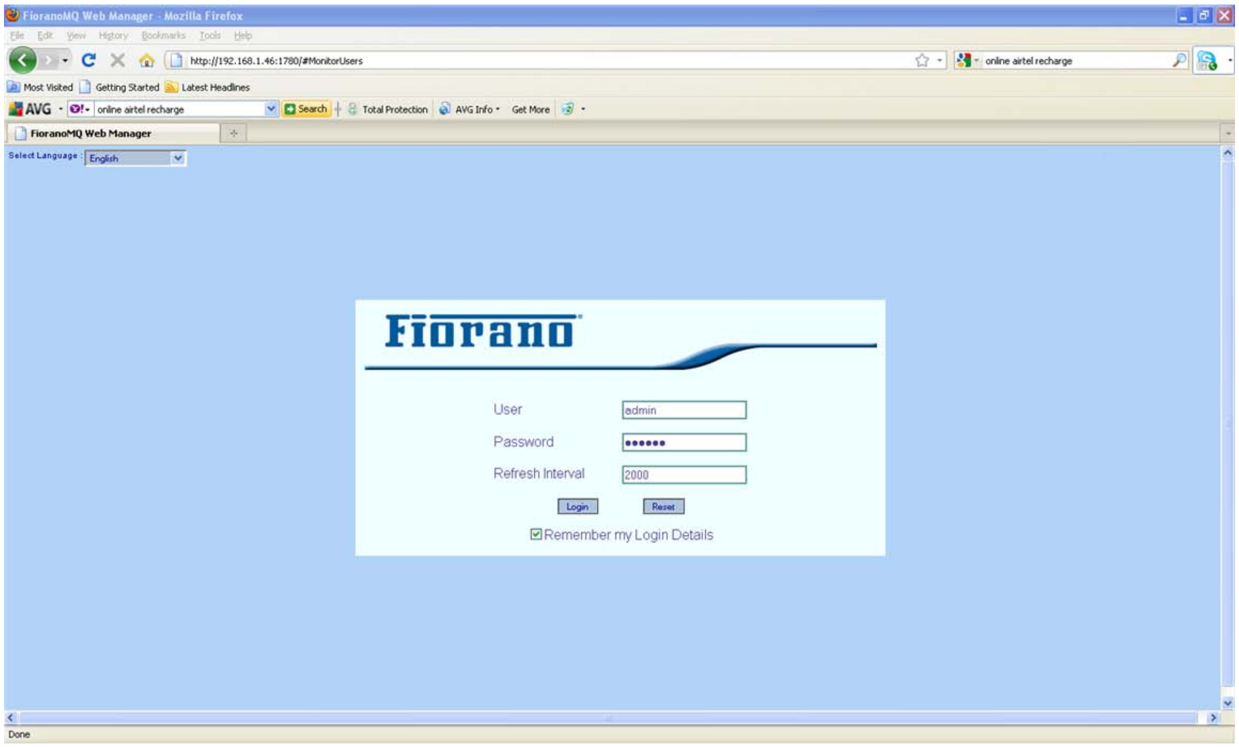
Note:

Field Name	Description

- The **EnableStart** and **PortNumber** must be configured before starting the FioranoMQ Server.
- All the parameters related to the WMT are present in the properties panel. These parameters may be changed in the online mode but require that the server is restarted once configurations are saved.

32.3 Connecting to Web Management Tool

To use FioranoMQ WMT, the latest version of the web browser (Internet Explorer, Mozilla Firefox, Safari, Opera) is needed . The WMT URL is http://ipaddress:jettyport. In this URL, the 'ipaddress' refers to the IP address or hostname (only one ipaddress is present) where the FioranoMQ Server is running and 'jettyport' refers to the port on which the WMT is running. If FioranoMQ server is running on IP address 192.168.1.46 and the WMT Server is running on port 1780, the URL will read http://192.168.1.46:1780.


FioranoMQ Web Manager - Mozilla Firefox
http://192.168.1.46:1780/#MonitorUsers
FioranoMQ Web Manager
Select Language : English
Fiorano
Login
Reset

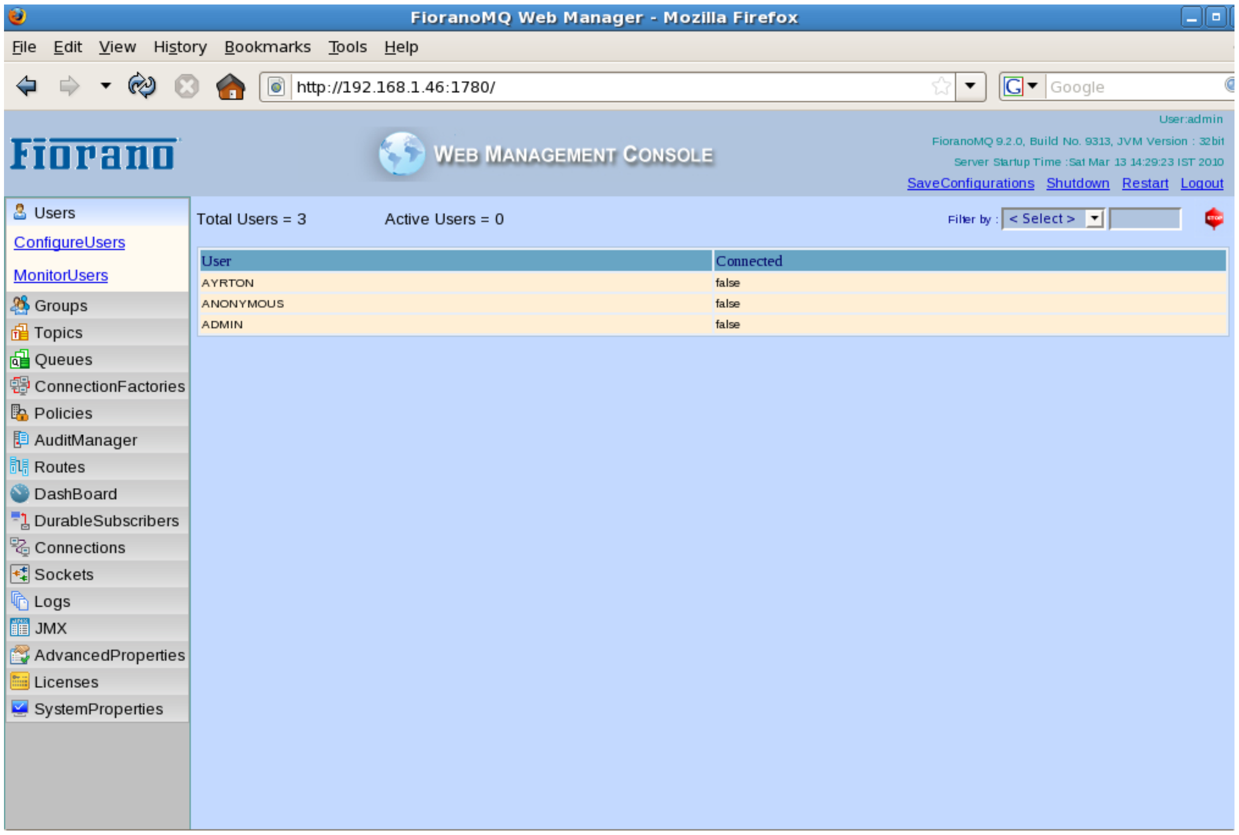
User	admin
Password	●●●●●●
Refresh Interval	2000
☑ Remember my Login Details

Figure 32.2: FioranoMQ Web Management Tool Homepage

The WMT homepage has three fields:

User	The User Name by which User will login tothe WMT. The User logging in must exist in the security Realm of FioranoMQ Server. The default User is ‘admin’.
Password	Password of the User Name by which the User connects to the WMT. The default password for the ‘admin’ is pre set.
Refresh Interval	Time Interval after which data is fetched from the Server. This Interval is measured in milli seconds. The value of this interval should be more than 1000 milliseconds. The default value is 2000 (which equivalent to two seconds). This interval should be sufficiently large to allow minimum CPU usage.

To perform certain operation administrator permissions are required.



User:admin
FioranoMQ 9.2.0, Build No. 9313, JVM Version : 32bit
Server Startup Time :Sat Mar 13 14:29:23 IST 2010

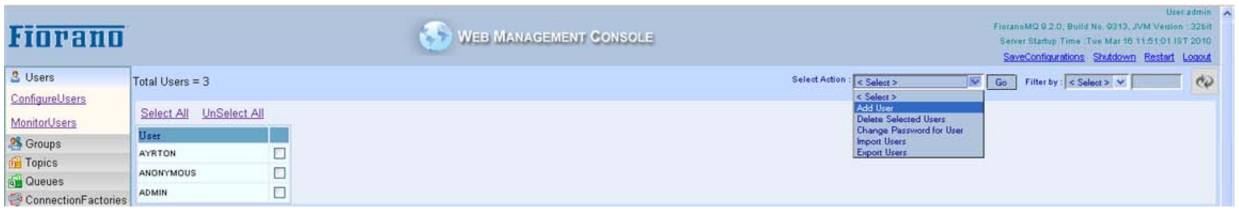
Total Users = 3 Active Users = 0

User	Connected
AYRTON	false
ANONYMOUS	false
ADMIN	false

Figure 32.3: MonitorUsers View

Once logged into WMT, the **MonitorUsers** may be view. All the tables present in the views can be sorted in both ascending and descending orders by clicking the header of the corresponding column. The images displayed in the column explain the order.

The **Filter** option can be viewed in the top right-hand corner. The Filter is used to filter tabular data based on the text present in textbox. Filtering is done based upon the filtering column selected in the menu.



User:admin
FioranoMQ 9.2.0, Build No. 9313, JVM Version : 32bit
Server Startup Time :Tue Mar 16 11:51:01 IST 2010

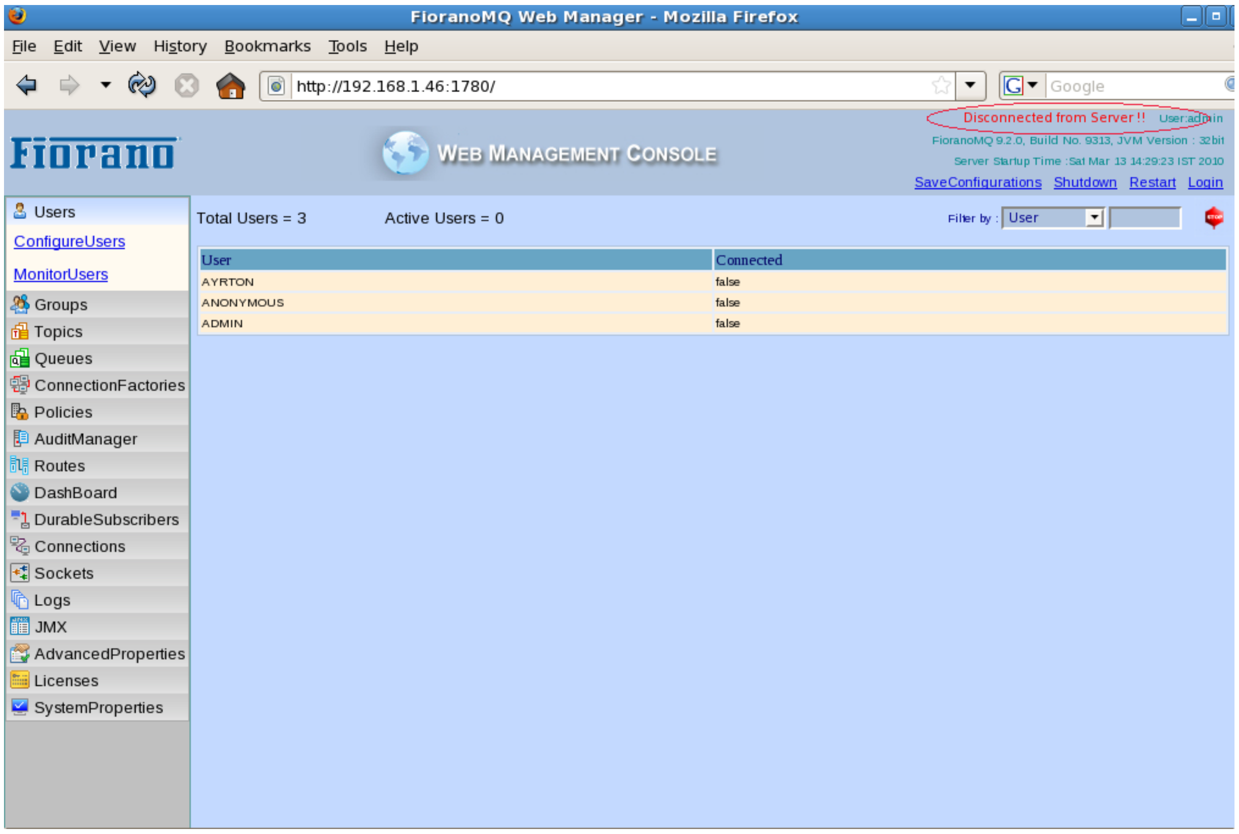
Total Users = 3

User	
AYRTON	☐
ANONYMOUS	☐
ADMIN	☐

Figure 32.4: Filter Options

All data present in the 'Monitor' view is refreshed automatically. 'Configure' views, on the other hand, need to be refreshed manually. For refreshing 'configure' view click on any Configure view and click again on the old configure view. Only **ConfigureFMQServer** present in JMX will be refreshed automatically. There is an image next to it. Data is fetched from the Server after every refresh interval. All tables get updated with the new values fetched. To stop refreshing this view, select the image and click the image to start refresh.

During interaction with the Server, if the Server shuts down due to unknown reasons, the WMT will display a warning to the User that reads 'disconnected from server'. When this warning is displayed, the **logout** prompt will change to the **login** prompt.



Disconnected from Server !! User:admin
FioranoMQ 9.2.0, Build No. 9313, JVM Version : 32bit
Server Startup Time :Sat Mar 13 14:29:23 IST 2010

Total Users = 3 Active Users = 0

User	Connected
AYRTON	false
ANONYMOUS	false
ADMIN	false

Figure 32.5: Server Disconnected Status

If the FioranoMQ Server running the User can click **login** to go directly to the page being viewed earlier. If the FioranoMQ Server is not running, the User will be redirected to the **login** page.

If the Server is restarted and the WMT did not interact with the Server, then clicking on any operation reconnects to Server. To avoid this/override this the User needs to click the same operation manually.

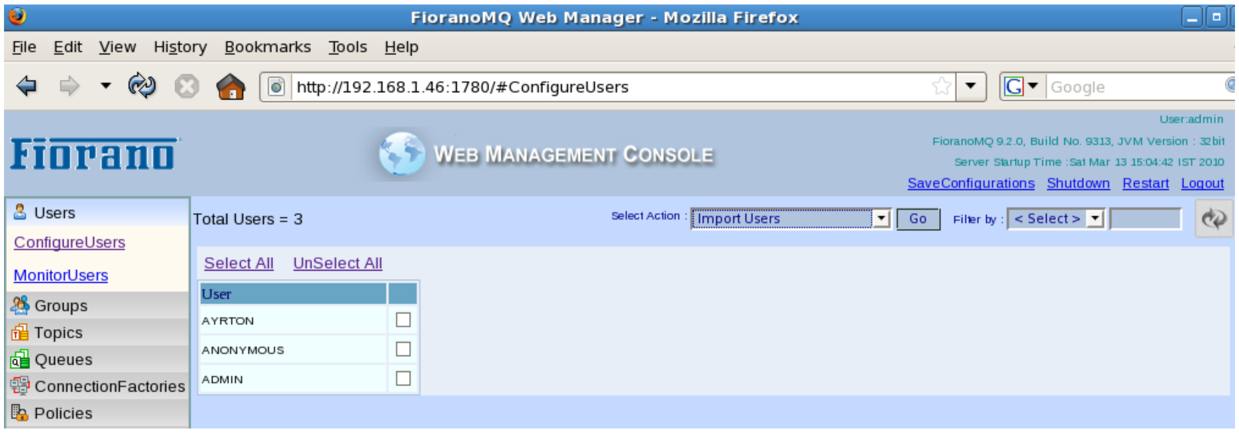
Four links at the top right-hand side top of the page may be viewed::

SaveConfigurations	SaveConfiguration saves all the modified properties to the FioranoMQ Server from the current browser page.
Shutdown	Shutdown closes the Server that is running. Do not connect any other application to FioranoMQ server before 'shutdown'.
Restart	Restart closes and restarts the Server. Do not connect any other application to FioranoMQ server before 'shutdown'. When the Server restarts, login from the browser window.
Logout	Logout disconnects the current page from the WMT Server.

There are 13 different views present in the WMT:

1. **Users**

   - **MonitorUsers** view contains all existing Users and their respective connection status. Users are set as connected an 'admin' Connection or JMX Connection is made or if the JMS Application is started.
   - **ConfigureUsers** view contains the function to modify Users (such as Add User and Delete Selected Users). The Import Users and Export Users enable to copying Users to another FioranoMQ Server. These actions are present in the **Select Action** drop-down menu. To use any one of the actions, select the action from the **Select Action** drop-down menu and click the **Go** button. These actions allow the export of all data to a standard file present on most machines. (Mapping needs to be done if file is present on another machine.) The data can be imported to any other Server that starts with FioranoMQ. Note: Import User and Export User functions can only be used by the 'admin' User.


FioranoMQ Web Manager - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://192.168.1.46:1780/#ConfigureUsers
Google
Fiorano
Web Management Console
User:admin
FioranoMQ 9.2.0, Build No. 9313, JVM Version : 32bit
Server Startup Time :Sat Mar 13 15:04:42 IST 2010
SaveConfigurations Shutdown Restart Logout
Users
ConfigureUsers
MonitorUsers
Groups
Topics
Queues
ConnectionFactories
Policies
Select Action : Import Users
Go
Filter by : < Select >

Total Users = 3

Select All UnSelect All

User	
AYRTON	☐
ANONYMOUS	☐
ADMIN	☐

Figure 32.6: Configure Users View

2. Groups

'Groups' contain information about all groups and its members. Groups can be added **using Add Groups** and deleted using **Delete Selected Groups**. For modifying members of an existing group, use Edit Members in Group after selecting the required group. These Actions are available in the **Select Action** drop-down menu. Select the action and click the **Go** button.

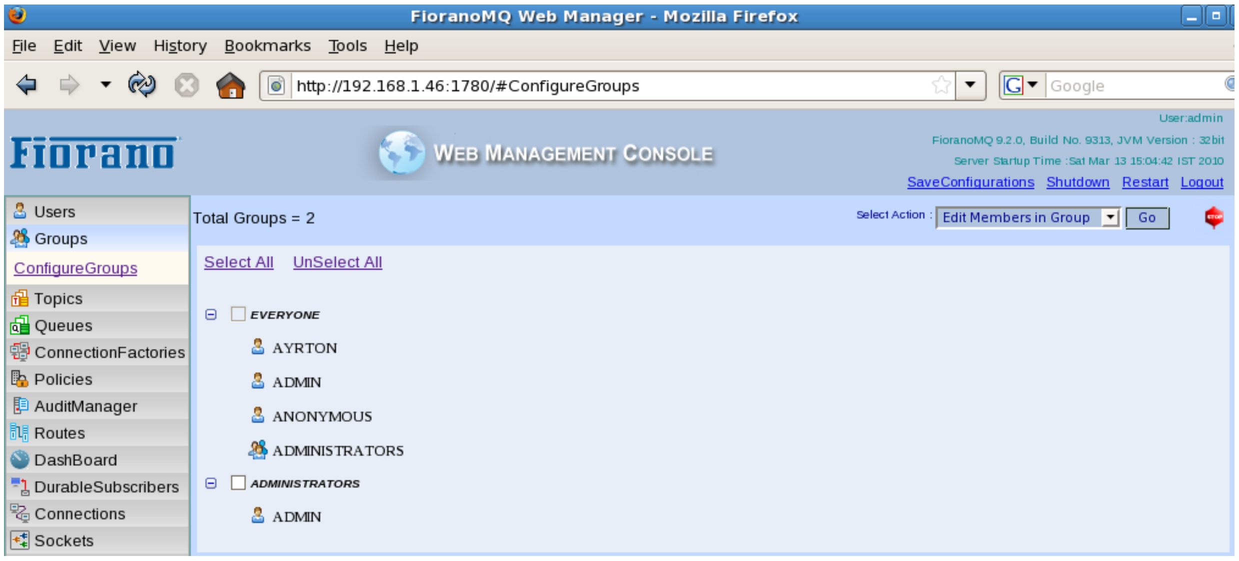

FioranoMQ Web Manager - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://192.168.1.46:1780/#ConfigureGroups
Fiorano
WEB MANAGEMENT CONSOLE
User:admin
FioranoMQ 9.2.0, Build No. 9313, JVM Version : 32bit
Server Startup Time :Sat Mar 13 15:04:42 IST 2010
SaveConfigurations Shutdown Restart Logout
Users
Groups
ConfigureGroups
Topics
Queues
ConnectionFactories
Policies
AuditManager
Routes
DashBoard
DurableSubscribers
Connections
Sockets
Total Groups = 2
Select Action : Edit Members in Group Go
Select All UnSelect All
EVERYONE
AYRTON
ADMIN
ANONYMOUS
ADMINISTRATORS
ADMINISTRATORS
ADMIN

Figure 32.7: GroupView

Once you click the Go button, a window pops up with existing members and possible options, if the selected action is **Edit Members in Group**. Select **Add Members** to see the existing members.

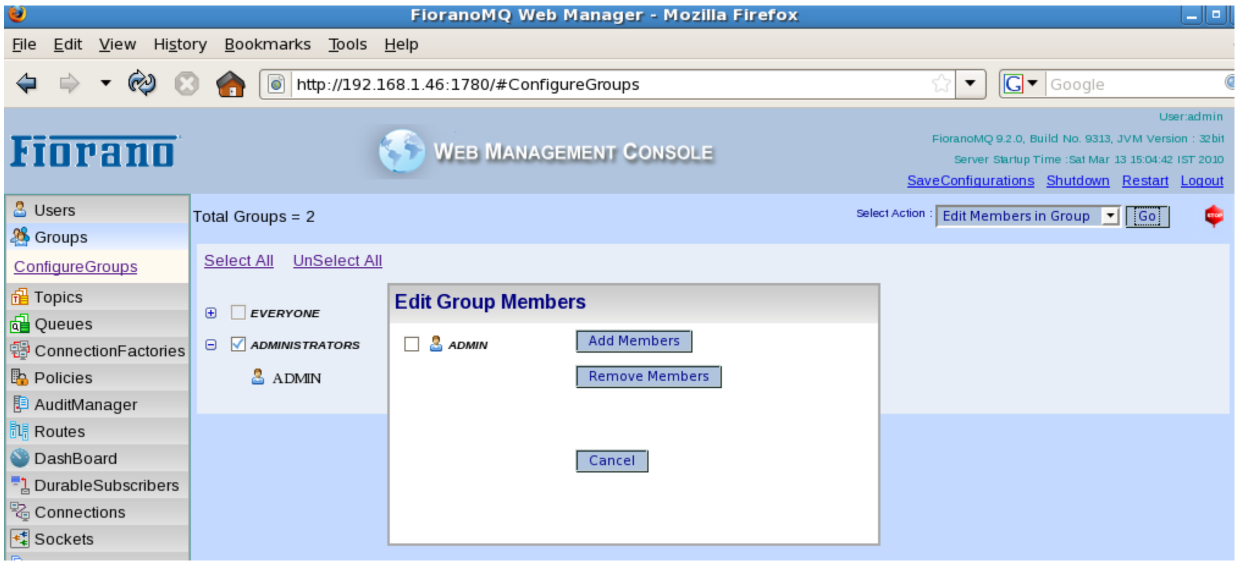
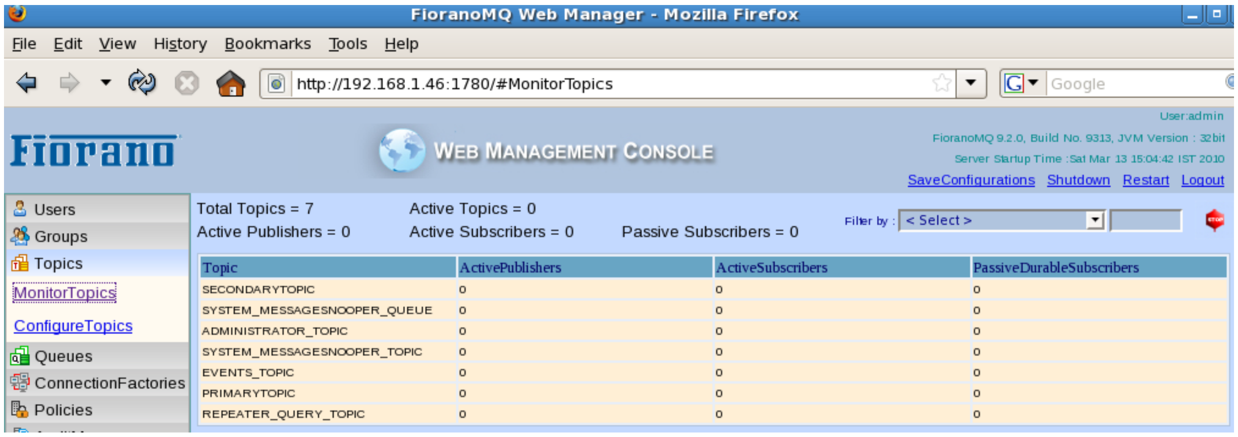

FioranoMQ Web Manager - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://192.168.1.46:1780/#ConfigureGroups
Fiorano
WEB MANAGEMENT CONSOLE
User:admin
FioranoMQ 9.2.0, Build No. 9313, JVM Version : 32bit
Server Startup Time :Sat Mar 13 15:04:42 IST 2010
SaveConfigurations Shutdown Restart Logout
Users
Groups
ConfigureGroups
Topics
Queues
ConnectionFactories
Policies
AuditManager
Routes
DashBoard
DurableSubscribers
Connections
Sockets
Total Groups = 2
Select Action : Edit Members in Group Go
Select All UnSelect All
EVERYONE
ADMINISTRATORS
ADMIN
Edit Group Members
ADMIN
Add Members
Remove Members
Cancel

Figure 32.8: Editing Group Members

3. **Topics**

Monitor Topics contains the information about existing topics. Summary is present at the top of the table. An **Active Topic** refers to a topics that contain either a publisher or a subscriber.

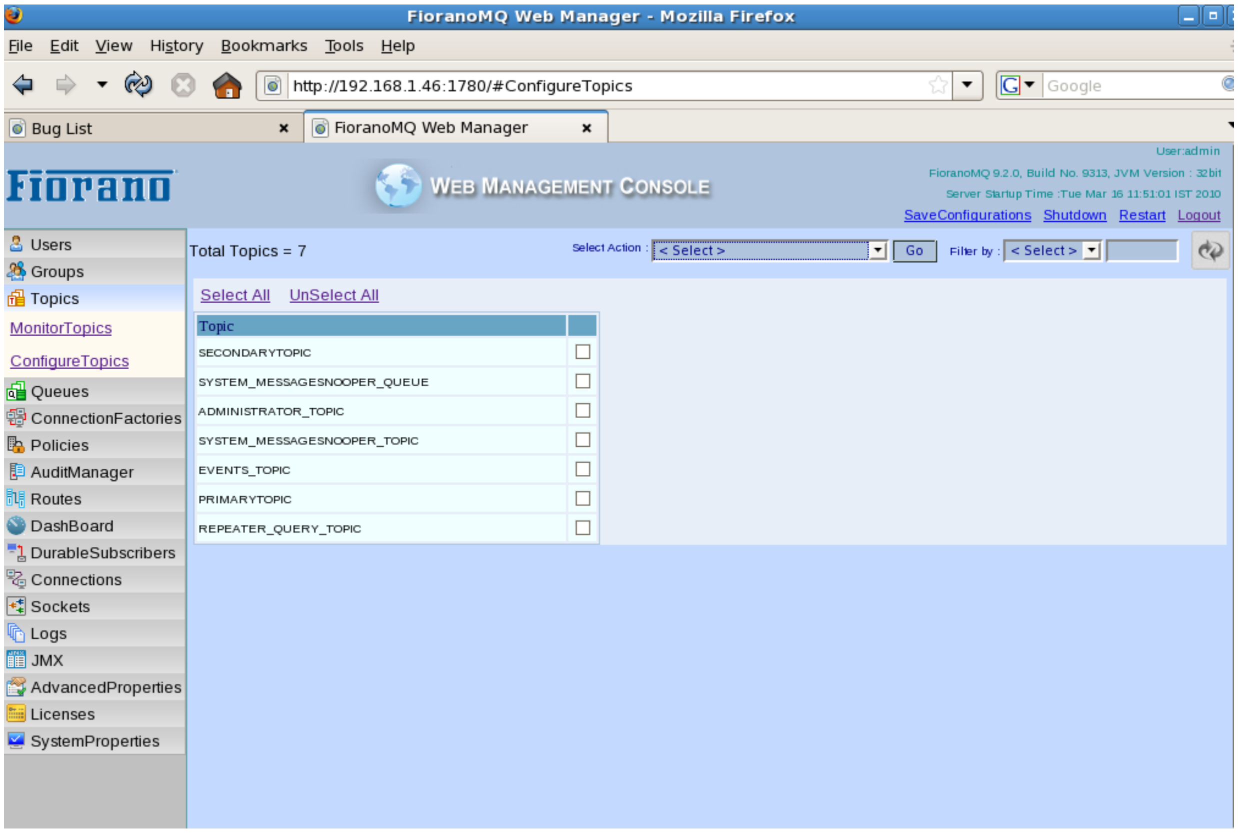

FioranoMQ Web Manager - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://192.168.1.46:1780/#MonitorTopics
Google
Fiorano
WEB MANAGEMENT CONSOLE
User:admin
FioranoMQ 9.2.0, Build No. 9313, JVM Version : 32bit
Server Startup Time :Sat Mar 13 15:04:42 IST 2010
SaveConfigurations Shutdown Restart Logout
Users
Groups
Topics
MonitorTopics
ConfigureTopics
Queues
ConnectionFactories
Policies
Filter by : < Select >

Total Topics = 7 Active Topics = 0
Active Publishers = 0 Active Subscribers = 0 Passive Subscribers = 0

Topic	ActivePublishers	ActiveSubscribers	PassiveDurableSubscribers
SECONDARYTOPIC	0	0	0
SYSTEM_MESSAGESNOOPER_QUEUE	0	0	0
ADMINISTRATOR_TOPIC	0	0	0
SYSTEM_MESSAGESNOOPER_TOPIC	0	0	0
EVENTS_TOPIC	0	0	0
PRIMARYTOPIC	0	0	0
REPEATER_QUERY_TOPIC	0	0	0

Figure 32.9: MonitorTopics View

Configure Topics allows all operations to be performed on topics such as **CreateTopic** and **DeleteSelectedTopics** . **EditTopicProperties** can be used to edit all properties related to a topic or the pubsub.

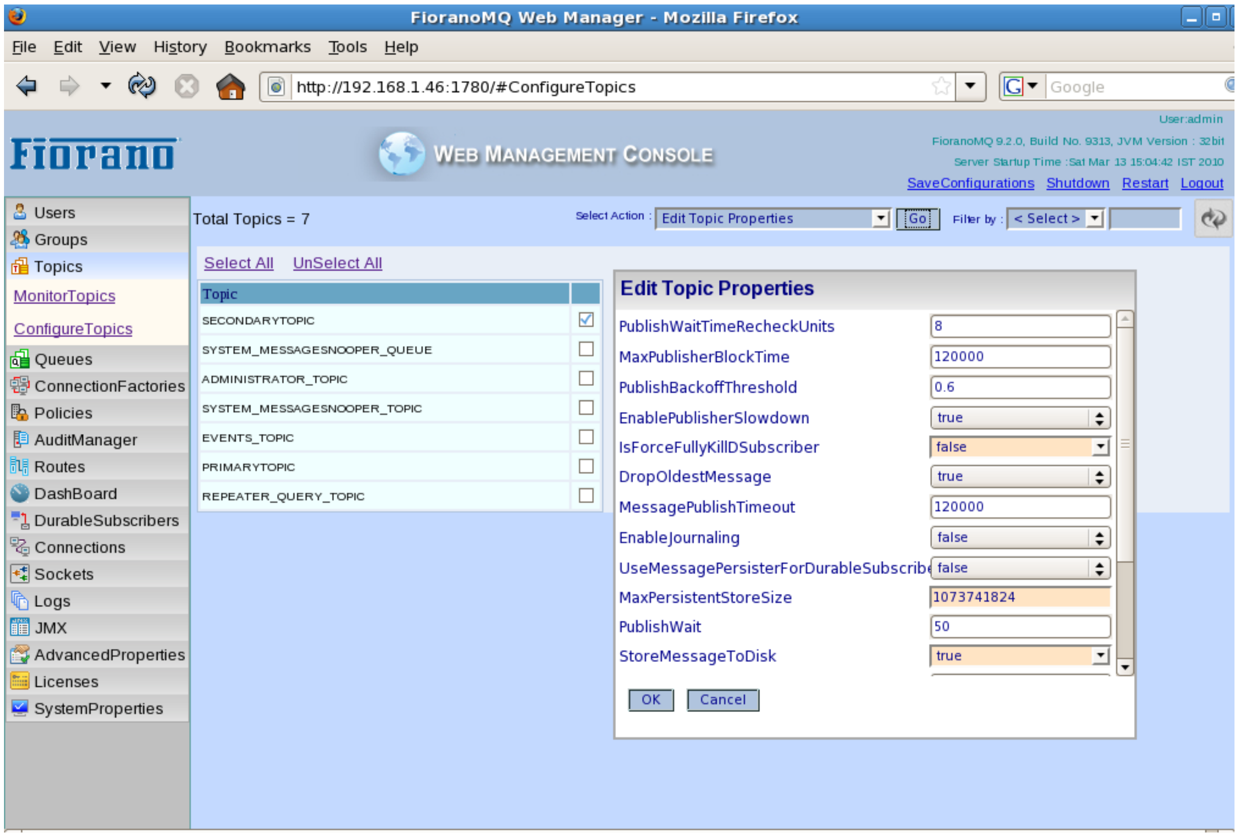


Total Topics = 7

Topic	
SECONDARYTOPIC	☐
SYSTEM_MESSAGESNOOPER_QUEUE	☐
ADMINISTRATOR_TOPIC	☐
SYSTEM_MESSAGESNOOPER_TOPIC	☐
EVENTS_TOPIC	☐
PRIMARYTOPIC	☐
REPEATER_QUERY_TOPIC	☐

Figure 32.10: ConfigureTopics View

By selecting the **EditTopicProperties** options from the **Select Action** drop-down menu and clicking the **Go** button, it is possible to view all properties and their corresponding values. Values with a different background color need to be saved. Once saved, the Server needs to be restarted to make them take effect. A description of the property that is selected is provided at the bottom of the browser. The mouse may be moved over the textbox to view the description of the property. **Export Topics** and **Import Topics** can be used to save topics to a standard file as well as to create all of them in a FioranoMQ server that is running. Import Topics and Export Topics can be used only by the 'admin' User.



http://192.168.1.46:1780/#ConfigureTopics

User:admin
FioranoMQ 9.2.0, Build No. 9313, JVM Version : 32bit
Server Startup Time :Sat Mar 13 15:04:42 IST 2010

Total Topics = 7

Select Action : Edit Topic Properties

Topic	
SECONDARYTOPIC	✓
SYSTEM_MESSAGESNOOPER_QUEUE	
ADMINISTRATOR_TOPIC	
SYSTEM_MESSAGESNOOPER_TOPIC	
EVENTS_TOPIC	
PRIMARYTOPIC	
REPEATER_QUERY_TOPIC	

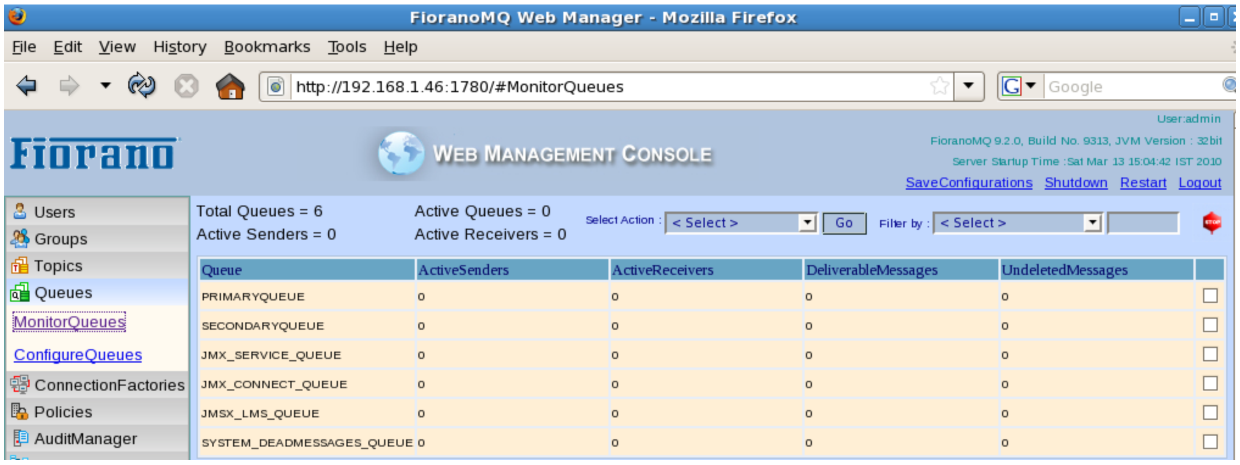
Edit Topic Properties

PublishWaitTimeRecheckUnits	8
MaxPublisherBlockTime	120000
PublishBackoffThreshold	0.6
EnablePublisherSlowdown	true
IsForceFullyKillDSubscriber	false
DropOldestMessage	true
MessagePublishTimeout	120000
EnableJournaling	false
UseMessagePersisterForDurableSubscribe	false
MaxPersistentStoreSize	1073741824
PublishWait	50
StoreMessageToDisk	true

Figure 32.11: Editing Topic Properties

4. **Queues**

Monitor Queues contains the information about existing queues. A summary is present at the top of the table. Here, **Active Queue** refers to the queues which contain either a sender or receiver. **DeliverableMessages** and **UndeletedMessages** do not update after regular intervals. To update these values select all the required queues or invoke **UpdateSelectedQueues** or **UpdateAllQueues** present in the **Select Action** drop-down menu.


FioranoMQ Web Manager - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://192.168.1.46:1780/#MonitorQueues
Google
Fiorano
WEB MANAGEMENT CONSOLE
User:admin
FioranoMQ 9.2.0, Build No. 9313, JVM Version : 32bit
Server Startup Time :Sat Mar 13 15:04:42 IST 2010
SaveConfigurations Shutdown Restart Logout
Users
Groups
Topics
Queues
MonitorQueues
ConfigureQueues
ConnectionFactories
Policies
AuditManager
Select Action : < Select > Go
Filter by : < Select >

Total Queues = 6 Active Queues = 0
Active Senders = 0 Active Receivers = 0

Queue	ActiveSenders	ActiveReceivers	DeliverableMessages	UndeletedMessages	
PRIMARYQUEUE	0	0	0	0	☐
SECONDARYQUEUE	0	0	0	0	☐
JMX_SERVICE_QUEUE	0	0	0	0	☐
JMX_CONNECT_QUEUE	0	0	0	0	☐
JMSX_LMS_QUEUE	0	0	0	0	☐
SYSTEM_DEADMESSAGES_QUEUE	0	0	0	0	☐

Figure 32.12: MonitorQueues View

Configure Queues allows all operations to be performed on queues such as **CreateQueue** and **DeleteSelectedQueues**. The **EditQueueProperties** can be used to edit all properties related to queue or ptp. By using EditQueueProperties option present under the **Select Action** drop-down menu it is possible to view all properties and their corresponding values. Values with different background color need to be saved. Once saved the Server needs to be restarted to make them take effect. A description of the property that is selected is provided atthe bottom of the browser. The mouse may be moved over the textbox to view the description of the property. You are notified if any property requires save configuration and server restart was modified. **Export Queues** and **Import Queues** can be used to save queues to a standard file as well as to create all of them in a FioranoMQ Server that is running. Import Queues and Export Queues can be used only by the 'admin' User.

The **Select Action** drop-down menu also contains operations like Browse Messages and Purge Messages. By using the Browse Messages option, all messages currently present on the selected queue may be viewed. The Purge Messages option can be used to delete all the messages present in the selected queue(s).

5. **ConnectionFactories**

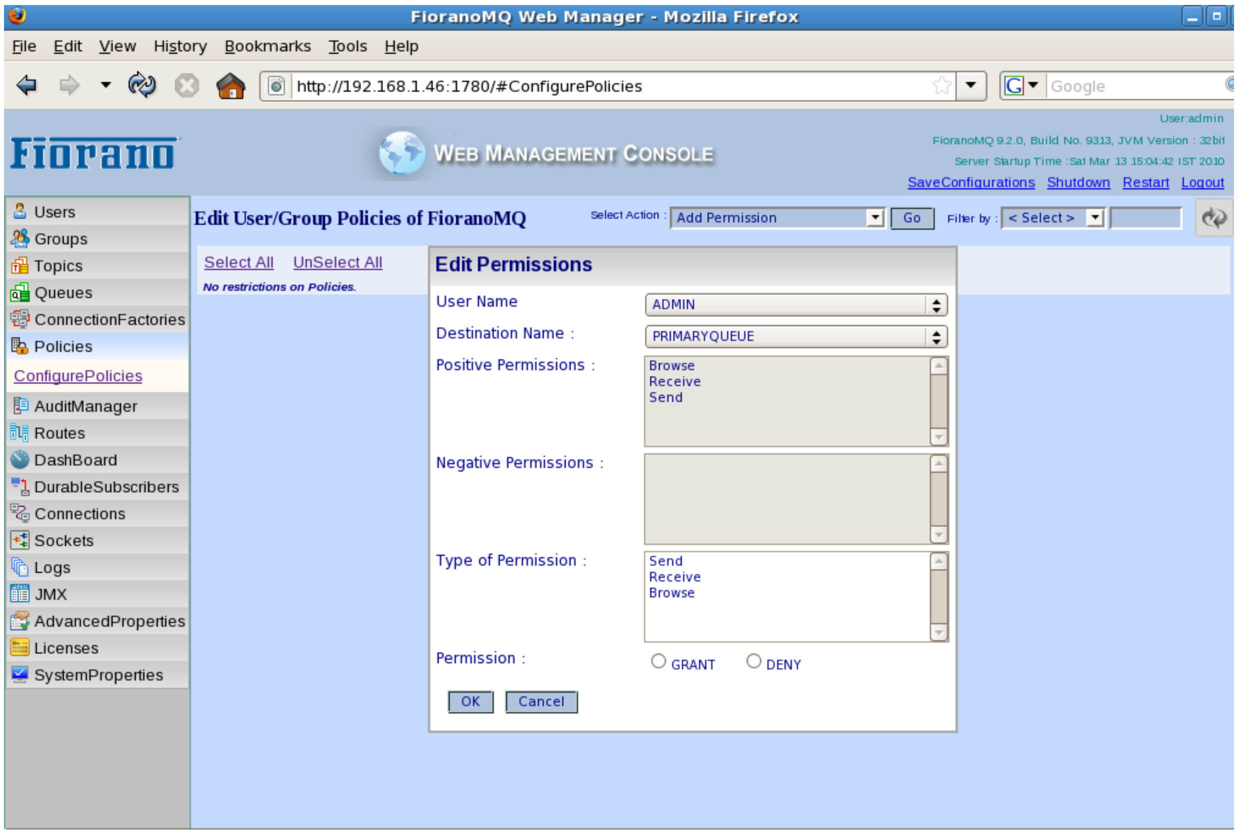
MonitorConnectionFactory lists all the connection factories along with their important properties (such as ConnectUrl, AutoUpdate, Type, Description). A summary is present at the top of the table.

ConfigureConnectionFactory allows various operations to be performed such as Create CF, Delete Selected CF, Edit CF Properties on Connection Factories. The Edit CF Properties can be used to edit all the properties related to the selected connection factory. The Create CF option present under the **Select Action** drop-down list can be used to create new Connection Factories and the Delete Selected CF can be used to delete existing Connection Factories.

Export and Import CF can be used to save Connection Factories to a standard file and to create all of them in a FioranoMQ Server that is running. Import and Export CF can be used only by the 'admin' User.

6. **Policies**

Policies allow users to change permissions on a specific destination for a specific principal. By default, the view will show **No restrictions on policies** if user had not modified any permission. If user wants to modify any particular permission then the user needs to use the **Add Permissions** option present in the Select Action drop-down menu. A pop-up window will appear along with UserName, Destination Name, Permission types, and new permission type. Once all required values are selected then user can click on **OK** button. The new modified permissions will be shown in the table.



http://192.168.1.46:1780/#ConfigurePolicies

User:admin

FioranoMQ 9.2.0, Build No. 9313, JVM Version : 32bit

Server Startup Time :Sat Mar 13 15:04:42 IST 2010

Edit User/Group Policies of FioranoMQ

Select Action : Add Permission

Filter by : < Select >

No restrictions on Policies.

Edit Permissions

User Name	ADMIN
Destination Name :	PRIMARYQUEUE
Positive Permissions :	Browse Receive Send
Negative Permissions :	
Type of Permission :	Send Receive Browse
Permission :	GRANT DENY

Figure 32.13: Editing policies of a destination

Click any row and view the permissions on a specific destination.

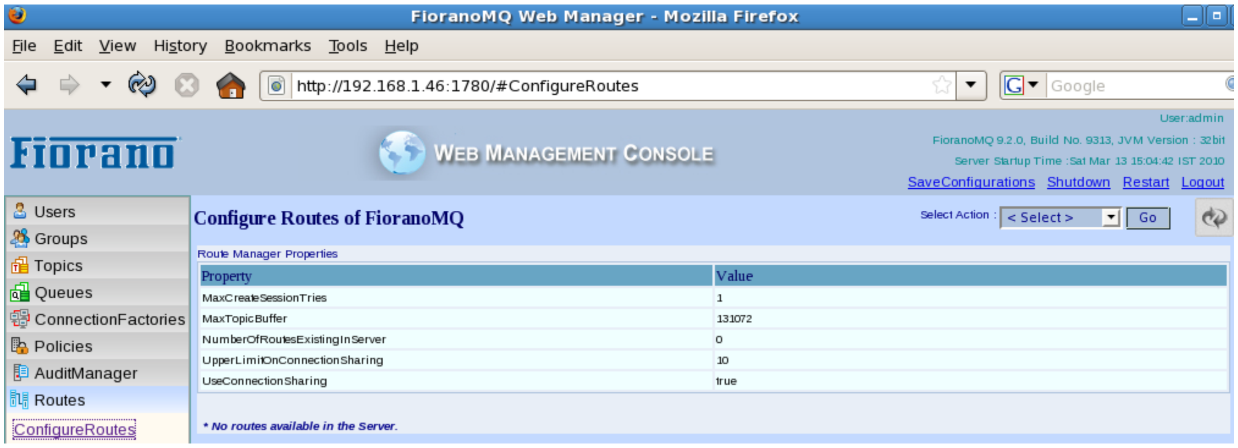
7. **Routes**

A Route enables the transport of messages between destinations. A message arriving on one destination can be made available to another provided a route exists between the two destinations.

ConfigureRoutes view contains information about the properties of the route managerThese are the properties of the individual routes present on the server.

The table displays various properties of the route manager (such as MaxTopicBuffer, NumberOfRoutesExistingInServer, MaxCreateSessionTries).

If there are any routes present on the Server, they will be displayed in the Route GUID box. If no routes are present, the message **No routes available in the Server** is displayed below the route manager properties table.


FioranoMQ Web Manager - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://192.168.1.46:1780/#ConfigureRoutes

WEB MANAGEMENT CONSOLE

User:admin
FioranoMQ 9.2.0, Build No. 9313, JVM Version : 32bit
Server Startup Time :Sat Mar 13 15:04:42 IST 2010
SaveConfigurations Shutdown Restart Logout

Users
Groups
Topics
Queues
ConnectionFactories
Policies
AuditManager
Routes
ConfigureRoutes

Configure Routes of FioranoMQ

Select Action : < Select > Go

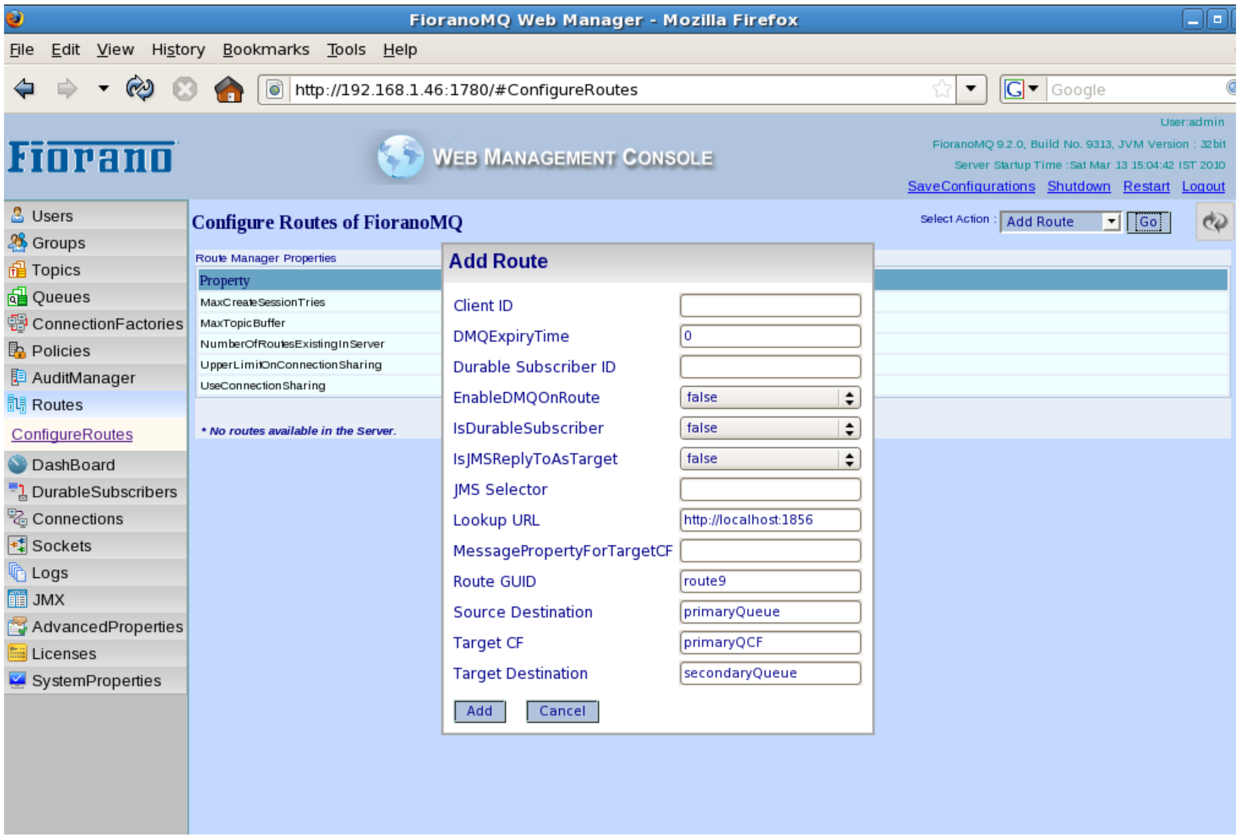
Route Manager Properties

Property	Value
MaxCreateSessionTries	1
MaxTopicBuffer	131072
NumberOfRoutesExistingInServer	0
UpperLimitOnConnectionSharing	10
UseConnectionSharing	true

* No routes available in the Server.

Figure 32.14: Route

To Add or Delete a route from the Server, the appropriate action must be chosen from the **Select Action** drop-down menu.


FioranoMQ Web Manager - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://192.168.1.46:1780/#ConfigureRoutes

WEB MANAGEMENT CONSOLE

User:admin
FioranoMQ 9.2.0, Build No. 9313, JVM Version : 32bit
Server Startup Time :Sat Mar 13 15:04:42 IST 2010
SaveConfigurations Shutdown Restart Logout

Users
Groups
Topics
Queues
ConnectionFactories
Policies
AuditManager
Routes
ConfigureRoutes
DashBoard
DurableSubscribers
Connections
Sockets
Logs
JMX
AdvancedProperties
Licenses
SystemProperties

Configure Routes of FioranoMQ

Select Action : Add Route Go

Route Manager Properties

Property
MaxCreateSessionTries
MaxTopicBuffer
NumberOfRoutesExistingInServer
UpperLimitOnConnectionSharing
UseConnectionSharing

* No routes available in the Server.

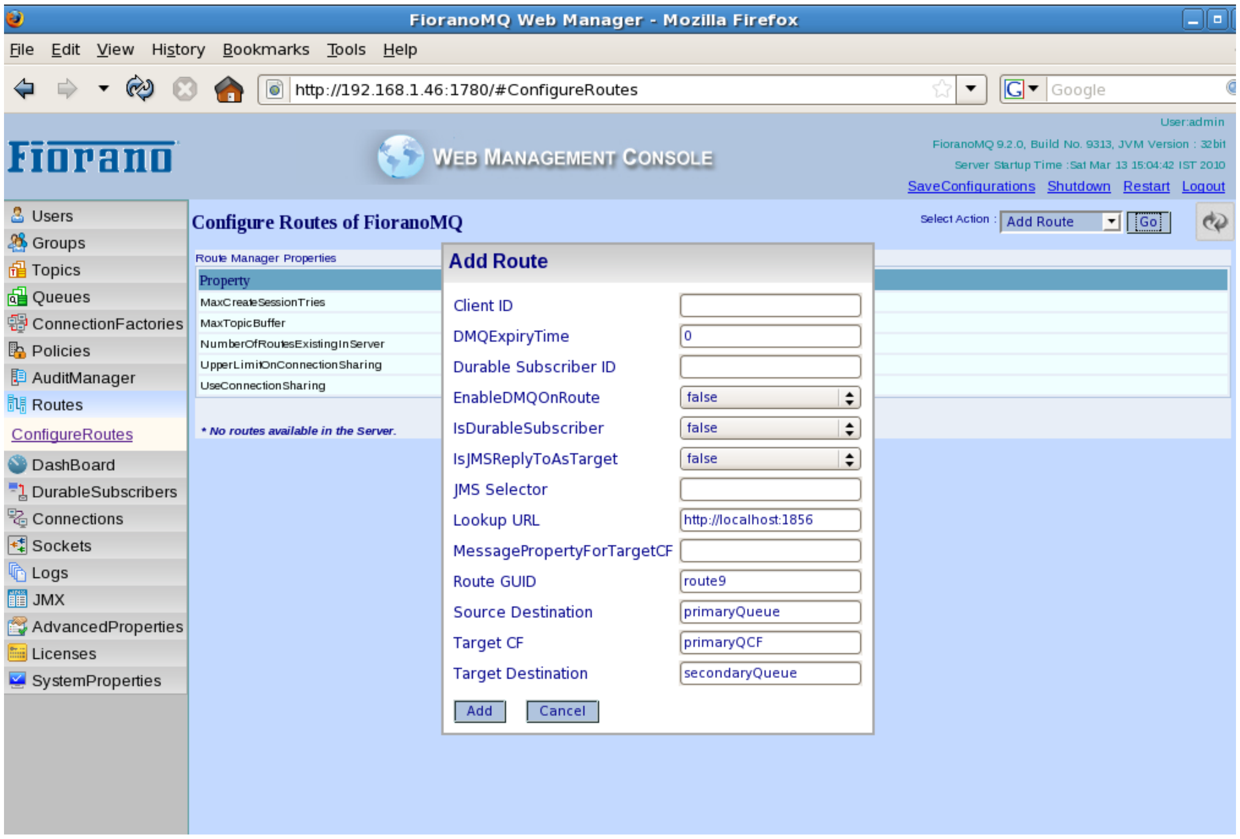
Add Route

Client ID	
DMQExpiryTime	0
Durable Subscriber ID	
EnableDMQOnRoute	false
IsDurableSubscriber	false
IsJMSReplyToAsTarget	false
JMS Selector	
Lookup URL	http://localhost:1856
MessagePropertyForTargetCF	
Route GUID	route9
Source Destination	primaryQueue
Target CF	primaryQCF
Target Destination	secondaryQueue

Add Cancel

Figure 32.15: Adding Route

On selecting any of the routes present in the Route GUID box the properties of the corresponding routes are displayed in the table below this list-box.


FioranoMQ Web Manager - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://192.168.1.46:1780/#ConfigureRoutes
Google
Fiorano
Web Management Console
User:admin
FioranoMQ 9.2.0, Build No. 9313, JVM Version : 32bit
Server Startup Time :Sat Mar 13 15:04:42 IST 2010
SaveConfigurations Shutdown Restart Logout
Users
Groups
Topics
Queues
ConnectionFactories
Policies
AuditManager
Routes
ConfigureRoutes
DashBoard
DurableSubscribers
Connections
Sockets
Logs
JMX
AdvancedProperties
Licenses
SystemProperties
Configure Routes of FioranoMQ
Select Action : Add Route Go
Route Manager Properties
Property
MaxCreateSessionTries
MaxTopicBuffer
NumberOfRoutesExistingInServer
UpperLimitOnConnectionSharing
UseConnectionSharing
* No routes available in the Server.
Add Route
Add
Cancel

Client ID	
DMQExpiryTime	0
Durable Subscriber ID	
EnableDMQOnRoute	false
IsDurableSubscriber	false
IsJMSReplyToAsTarget	false
JMS Selector	
Lookup URL	http://localhost:1856
MessagePropertyForTargetCF	
Route GUID	route9
Source Destination	primaryQueue
Target CF	primaryQCF
Target Destination	secondaryQueue

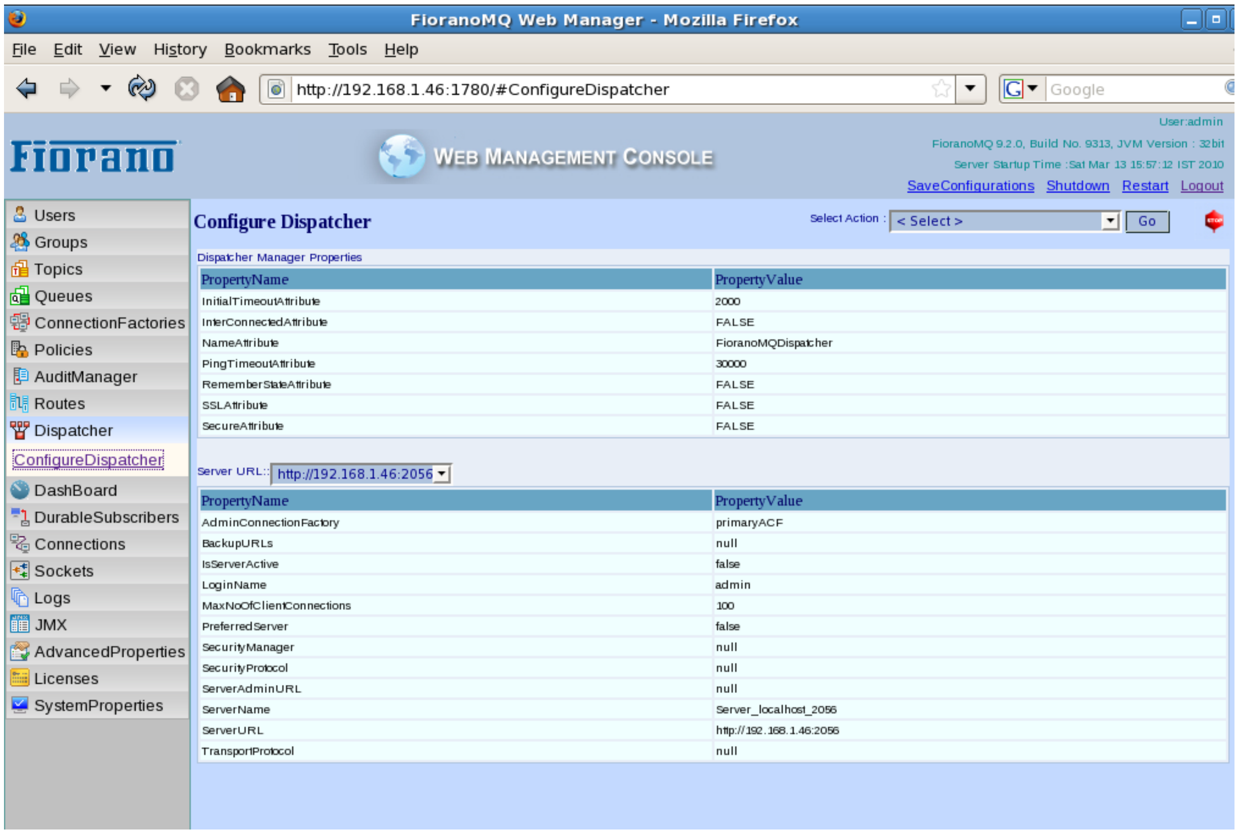
Figure 32.16: Route Added

8. **Dispatcher**

This tab is available only when logged into the web console of a dispatcher enabled Server.

A Dispatcher distributes the messages received to different Servers that are connected to it. The Dispatcher performs the load balancing function of a network connected to FioranoMQ Servers.

ConfigureDispatcher view contains information about the properties of the Dispatcher Manager and the properties of the Servers to which it dispatches messages. The Server URL list-box contains the list of server URLs to which the Dispatcher can dispatch messages. On selecting a URL, its properties are displayed on the table as shown in the figure 32.17.



FioranoMQ Web Manager - Mozilla Firefox

http://192.168.1.46:1780/#ConfigureDispatcher

User:admin

FioranoMQ 9.2.0, Build No. 9313, JVM Version : 32bit

Server Startup Time :Sat Mar 13 15:57:12 IST 2010

Configure Dispatcher

Select Action : < Select >

Dispatcher Manager Properties

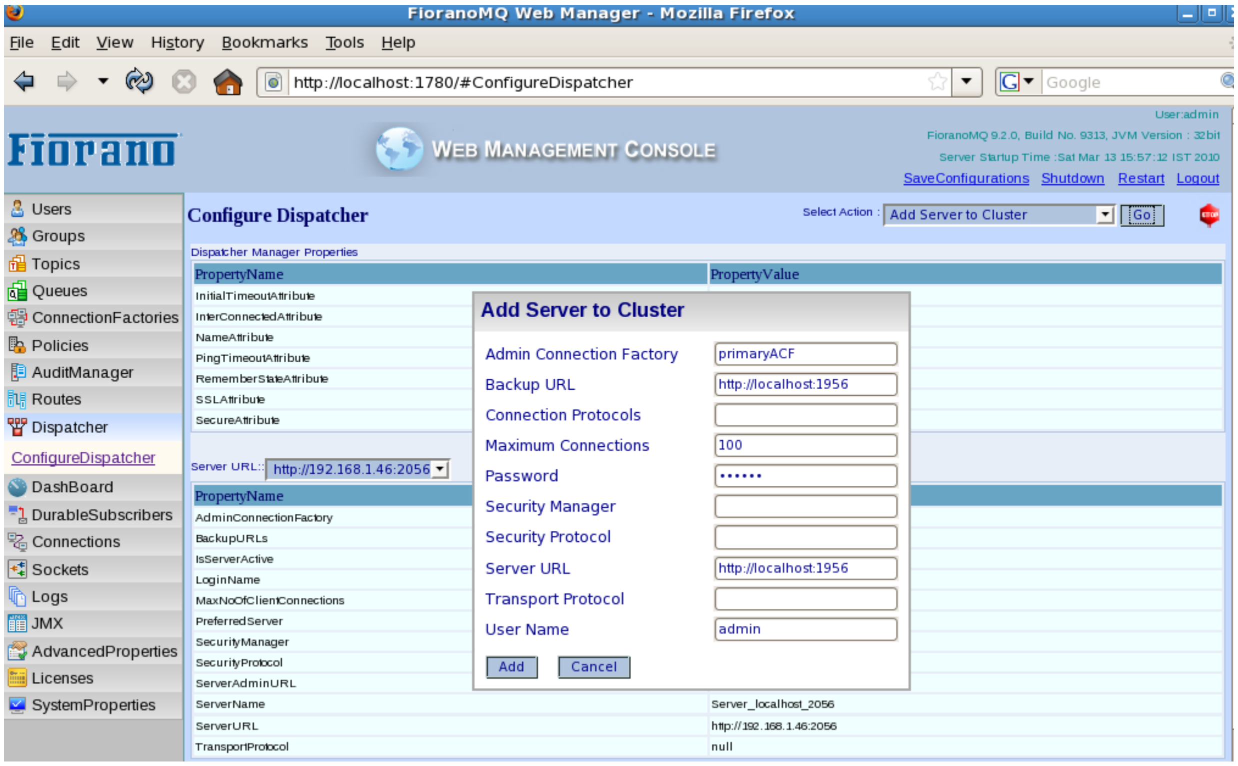
PropertyName	PropertyValue
InitialTimeoutAttribute	2000
InterConnectedAttribute	FALSE
NameAttribute	FioranoMQDispatcher
PingTimeoutAttribute	30000
RememberStateAttribute	FALSE
SSLAttribute	FALSE
SecureAttribute	FALSE

Server URL:: http://192.168.1.46:2056

PropertyName	PropertyValue
AdminConnectionFactory	primaryACF
BackupURLs	null
IsServerActive	false
LoginName	admin
MaxNoOfClientConnections	100
PreferredServer	false
SecurityManager	null
SecurityProtocol	null
ServerAdminURL	null
ServerName	Server_localhost_2056
ServerURL	http://192.168.1.46:2056
TransportProtocol	null

Figure 32.17: Dispatcher

To Add or Delete a Dispatcher Server, the appropriate action must be chosen from the **Select Action** drop-down menu..



FioranoMQ Web Manager - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://localhost:1780/#ConfigureDispatcher

Fiorano WEB MANAGEMENT CONSOLE

User:admin
FioranoMQ 9.2.0, Build No. 9313, JVM Version : 32bit
Server Startup Time :Sat Mar 13 15:57:12 IST 2010
SaveConfigurations Shutdown Restart Logout

Users
Groups
Topics
Queues
ConnectionFactories
Policies
AuditManager
Routes
Dispatcher
ConfigureDispatcher
DashBoard
DurableSubscribers
Connections
Sockets
Logs
JMX
AdvancedProperties
Licenses
SystemProperties

Configure Dispatcher

Select Action : Add Server to Cluster Go

Dispatcher Manager Properties

PropertyName	PropertyValue
InitialTimeoutAttribute	
InterConnectedAttribute	
NameAttribute	
PingTimeoutAttribute	
RememberStateAttribute	
SSLAttribute	
SecureAttribute	

Server URL:: http://192.168.1.46:2056

PropertyName	
AdminConnectionFactory	
BackupURLs	
IsServerActive	
LoginName	
MaxNoOfClientConnections	
PreferredServer	
SecurityManager	
SecurityProtocol	
ServerAdminURL	
ServerName	Server_localhost_2056
ServerURL	http://192.168.1.46:2056
TransportProtocol	null

Add Server to Cluster

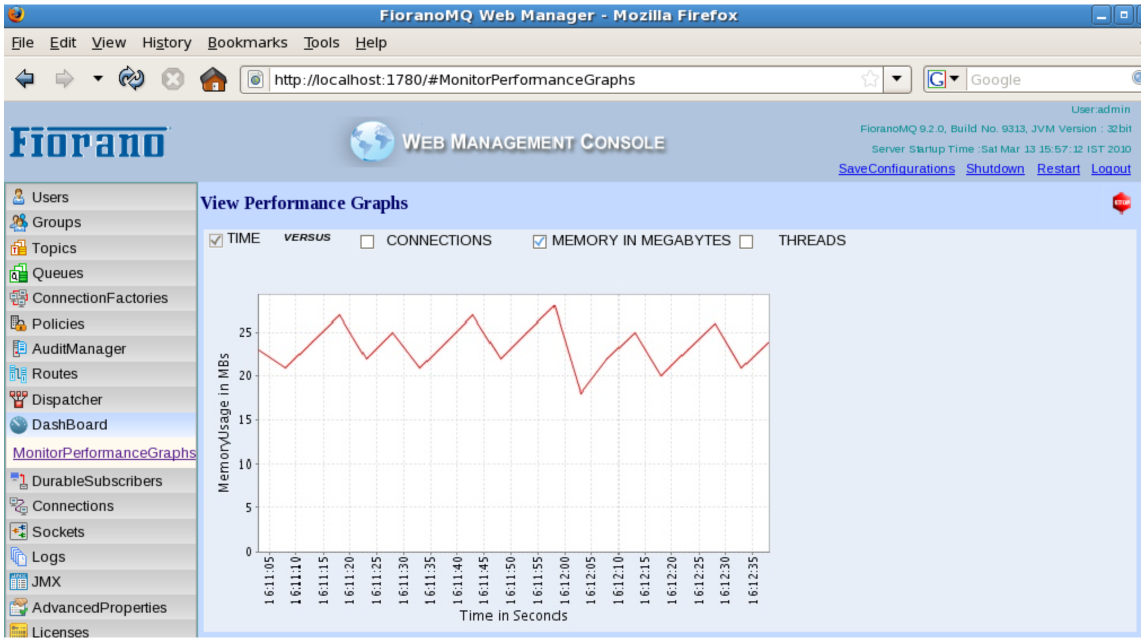
Admin Connection Factory	primaryACF
Backup URL	http://localhost:1956
Connection Protocols	
Maximum Connections	100
Password	••••••
Security Manager	
Security Protocol	
Server URL	http://localhost:1956
Transport Protocol	
User Name	admin

Add Cancel

Figure 32.18: AddDispatcher Server

9. DashBoard

DashBoard shows the graphs between Time vs Threads, Connections and MemoryThe entry for which graphs are required needs to be selected.. The **Remove** selection, removes a graph from view.



FioranoMQ Web Manager - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://localhost:1780/#MonitorPerformanceGraphs

Fiorano WEB MANAGEMENT CONSOLE

User:admin
FioranoMQ 9.2.0, Build No. 9313, JVM Version : 32bit
Server Startup Time :Sat Mar 13 15:57:12 IST 2010
SaveConfigurations Shutdown Restart Logout

Users
Groups
Topics
Queues
ConnectionFactories
Policies
AuditManager
Routes
Dispatcher
DashBoard
MonitorPerformanceGraphs
DurableSubscribers
Connections
Sockets
Logs
JMX
AdvancedProperties
Licenses

View Performance Graphs

[✓] TIME *VERSUS* [ ] CONNECTIONS [✓] MEMORY IN MEGABYTES [ ] THREADS

Figure 32.19: Displaying a graph of Time Vs Memory

10. **DurableSubscribers**

Monitor DurableSubscriber view contains all existing durable subscribers listed along with corresponding information. A summary is present at the top of this window. To update deliverable messages, click on the header.

Configure DurableSubscriber view contains all durable subscriber names in the form of subscriberID::clientID. Messages can be unsubscribe or purged messages for passive durable subscribers from this view. Any operations involving active durable subscribers throw a warning message.

11. **Connections**

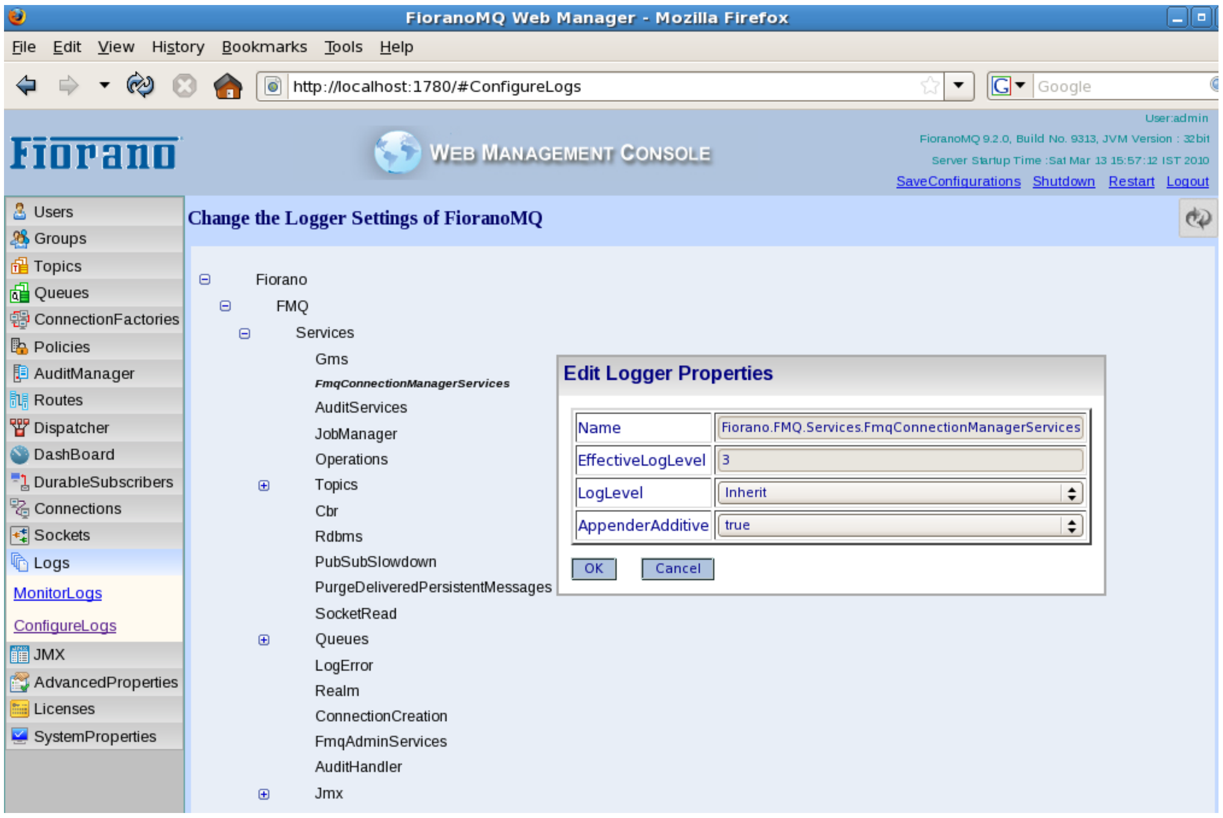
Monitor Connections lists all type of connections along their relevant properties (such as creation time, client address and user name of the person creating the connection). A summary is present at the top of this window.

Configure Connections allows the disconnection of a connection based on the clientID.

12. **Logs**

Monitor Logs allows the user to fetch details of logs, view log details and clear log details.

Configure Logs lists all those logged into the FioranoMQ Server. You can select any particular log and change its level.



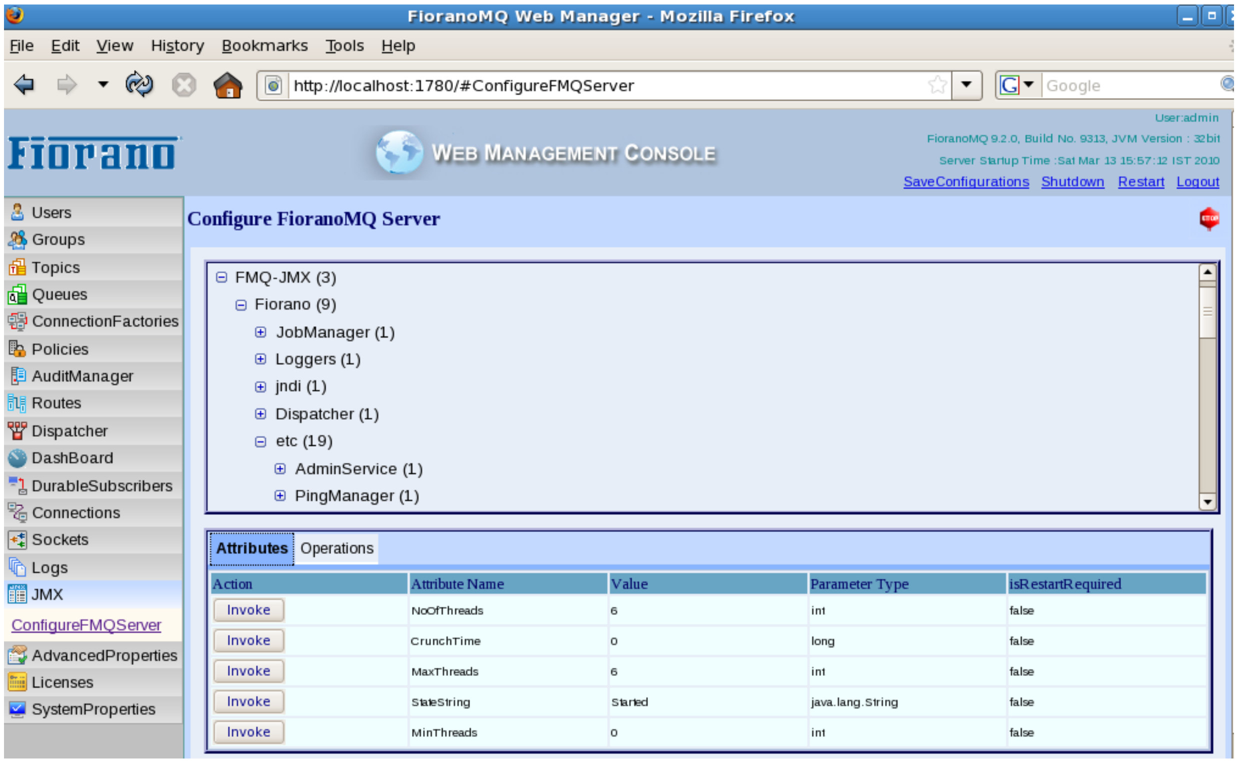
Name	Fiorano.FMQ.Services.FmqConnectionManagerServices
EffectiveLogLevel	3
LogLevel	Inherit
AppenderAdditive	true

Figure 32.20: Changing log level of a particular logger

13. **JMX**

This view lists all the Mbeans along with their attributes and operations. Mbeans are listed in this hierarchical view with a brown color bullet.

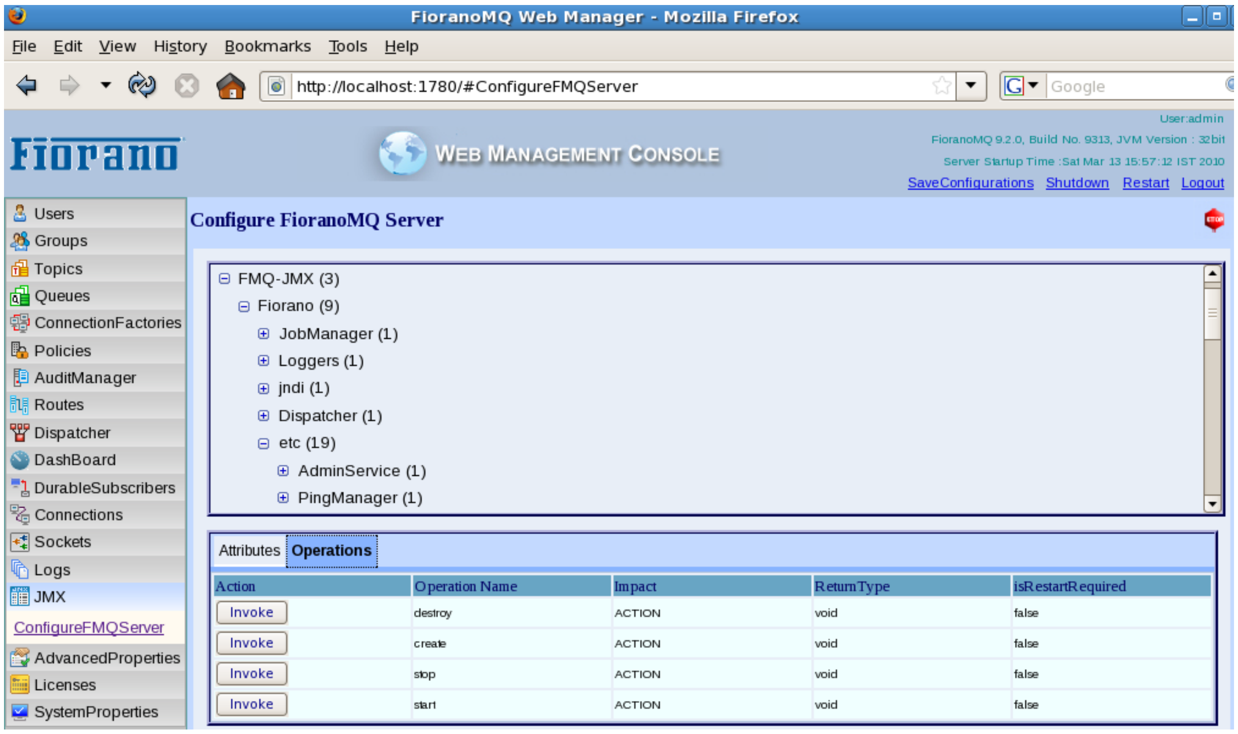
When a particular Mbean is selected, a table containing two tabs is displayed. The **Attributes** tab lists all the attributes of the Mbeans along with their values, parameter types and whether a restart of the Server is required for changes to take effect.



Action	Attribute Name	Value	Parameter Type	isRestartRequired
Invoke	NoOfThreads	6	int	false
Invoke	CrunchTime	0	long	false
Invoke	MaxThreads	6	int	false
Invoke	StateString	Started	java.lang.String	false
Invoke	MinThreads	0	int	false

Figure 32.21: Attribute View of JMX

The **Operations** tab lists the operations present in that Mbean enabling the User to invoke these operations.



Action	Operation Name	Impact	ReturnType	isRestartRequired
Invoke	destroy	ACTION	void	false
Invoke	create	ACTION	void	false
Invoke	stop	ACTION	void	false
Invoke	start	ACTION	void	false

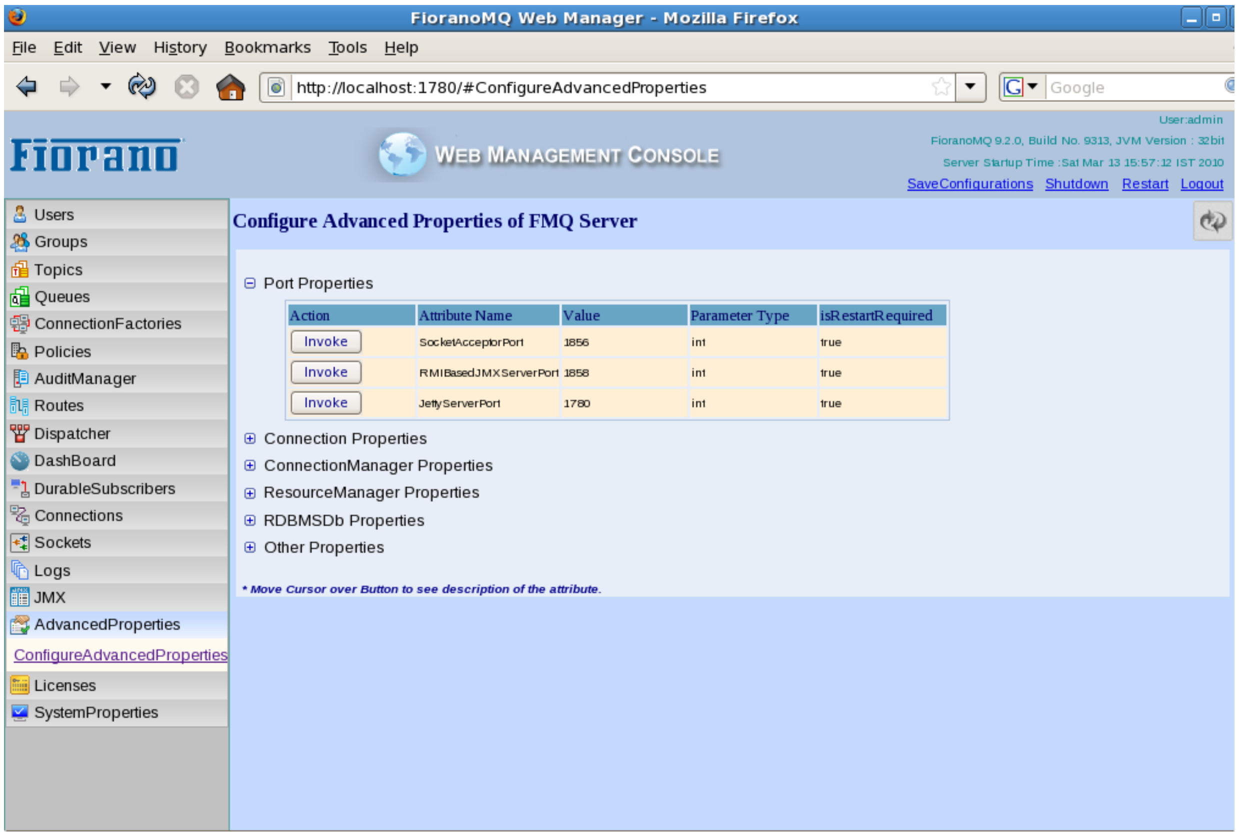
Figure 32.22: Operations View of JMX

Clicking on **SHOW_OPERATIONS** lists the operations present in that MBean enabling the User to invoke the operation.

14. **AdvancedProperties**

This view contains frequently used properties and their values. Categorization isbased on the values.

For example, the first category, **Port properties** contain Socket Acceptor port, RMIBasedJMXServerPort and JettyServer port. These are the three ports which need to be unique forrunning a server on the same IPAddress. These values can be modified.


FioranoMQ Web Manager - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost:1780/#ConfigureAdvancedProperties
Google
Fiorano
Web Management Console
User:admin
FioranoMQ 9.2.0, Build No. 9313, JVM Version : 32bit
Server Startup Time :Sat Mar 13 15:57:12 IST 2010
SaveConfigurations Shutdown Restart Logout
Users
Groups
Topics
Queues
ConnectionFactories
Policies
AuditManager
Routes
Dispatcher
DashBoard
DurableSubscribers
Connections
Sockets
Logs
JMX
AdvancedProperties
ConfigureAdvancedProperties
Licenses
SystemProperties

Configure Advanced Properties of FMQ Server

Port Properties

Action	Attribute Name	Value	Parameter Type	isRestartRequired
Invoke	SocketAcceptorPort	1856	int	true
Invoke	RMIBasedJMXServerPort	1858	int	true
Invoke	JettyServerPort	1780	int	true

Connection Properties
ConnectionManager Properties
ResourceManager Properties
RDBMSDb Properties
Other Properties

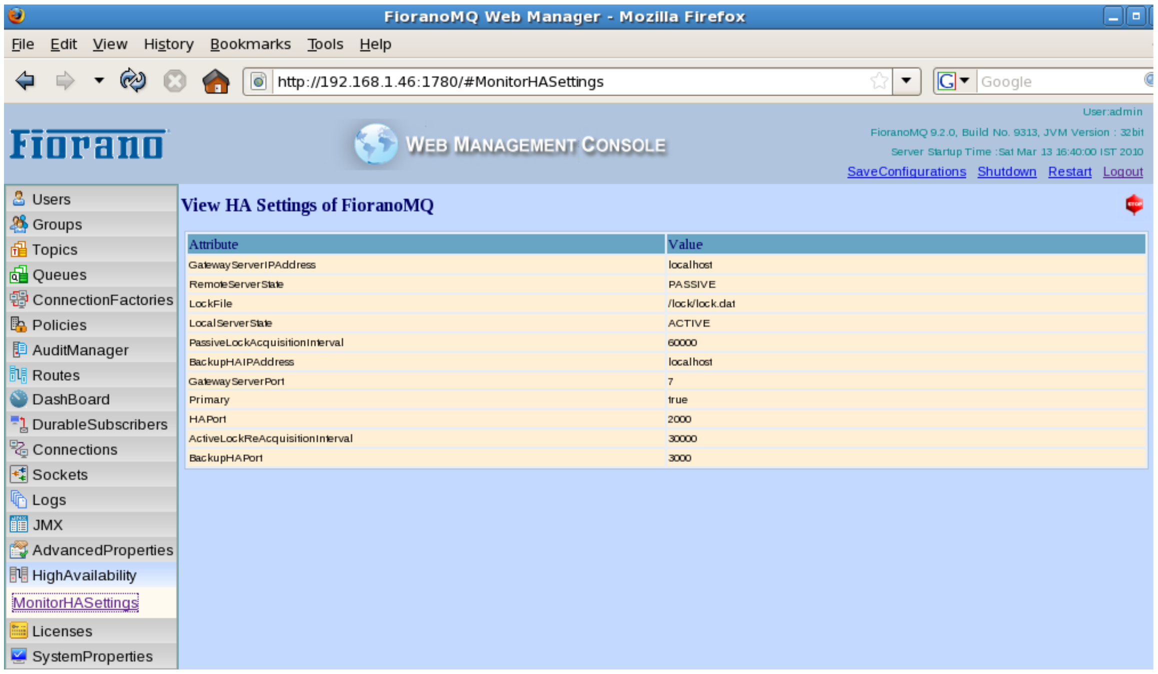
* Move Cursor over Button to see description of the attribute.

Figure 32.23: AdvancedProperties View

15. **HighAvailability**

This view contains properties related to the HA Servers. When using theStandAlone Server this view shows a message with: **The Server you logged in is not running in HA Mode.**

If the HA server is running this view displays all relevant properties for HA.


FioranoMQ Web Manager - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://192.168.1.46:1780/#MonitorHASettings
Google
Fiorano
WEB MANAGEMENT CONSOLE
User:admin
FioranoMQ 9.2.0, Build No. 9313, JVM Version : 32bit
Server Startup Time :Sat Mar 13 16:40:00 IST 2010
SaveConfigurations Shutdown Restart Logout
Users
Groups
Topics
Queues
ConnectionFactories
Policies
AuditManager
Routes
DashBoard
DurableSubscribers
Connections
Sockets
Logs
JMX
AdvancedProperties
HighAvailability
MonitorHASettings
Licenses
SystemProperties

View HA Settings of FioranoMQ

Attribute	Value
GatewayServerIPAddress	localhost
RemoteServerState	PASSIVE
LockFile	/lock/lock.dat
LocalServerState	ACTIVE
PassiveLockAcquisitionInterval	60000
BackupHAIPAddress	localhost
GatewayServerPort	7
Primary	true
HAPort	2000
ActiveLockReAcquisitionInterval	30000
BackupHAPort	3000

Figure 32.24: HA view

16. **StandAloneBridge**

This tab is available only when logged into the web console of a Server running on the StandAloneBridge profile.

A Bridge transmits a message received on one queue (source queue) to another queue (target queue).

ConfigureBridge view lists all the Bridges currently available on the server. The **Select Action** drop-down menu contains various options , such as:

- View Bridge Attributes
- Add Link
- Delete Link
- View Link Configuration
- Edit Link Configuration
- Add Channel
- Delete Channels
- View Channel Configuration

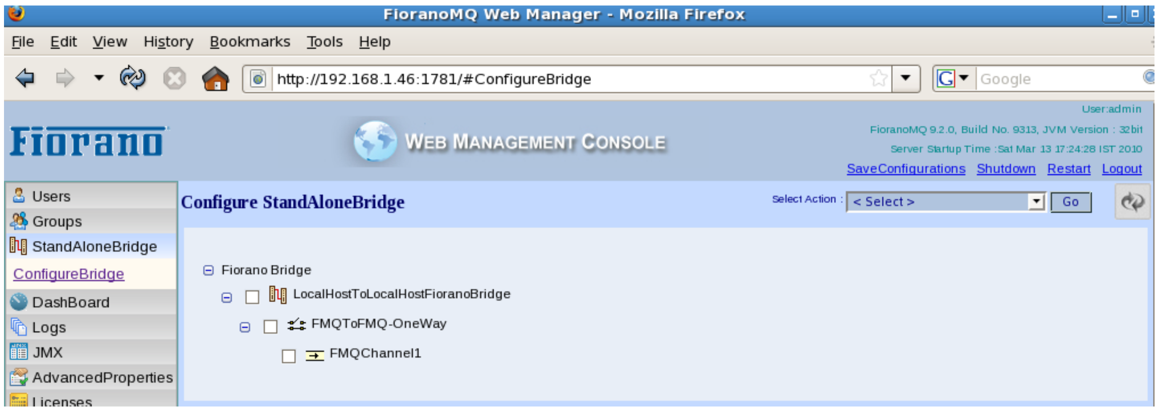

FioranoMQ Web Manager - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://192.168.1.46:1781/#ConfigureBridge
Fiorano
WEB MANAGEMENT CONSOLE
User:admin
FioranoMQ 9.2.0, Build No. 9313, JVM Version : 32bit
Server Startup Time :Sat Mar 13 17:24:28 IST 2010
SaveConfigurations Shutdown Restart Logout
Users
Groups
StandAloneBridge
ConfigureBridge
DashBoard
Logs
JMX
AdvancedProperties
Licenses
Configure StandAloneBridge
Select Action : < Select > Go
Fiorano Bridge
LocalHostToLocalHostFioranoBridge
FMQToFMQ-OneWay
FMQChannel1

Figure 32.25: StandAloneBridge

AddLink Attributes

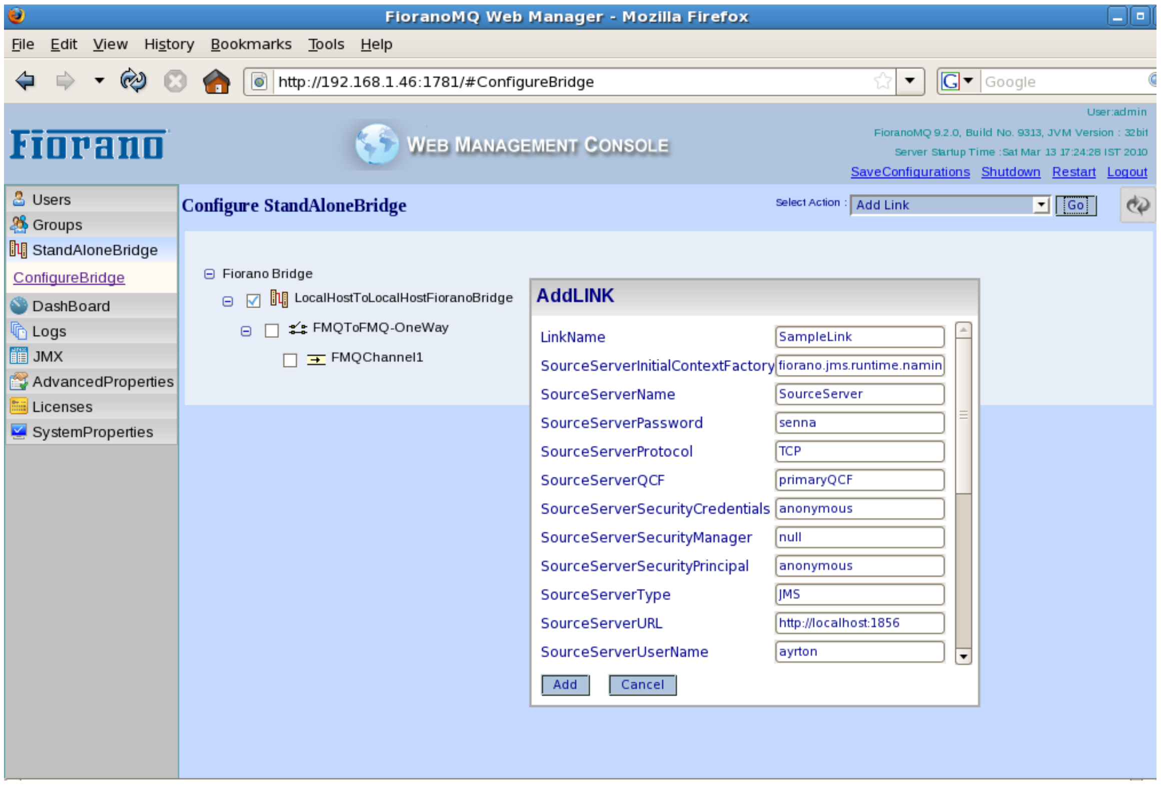
To add a link select Action list-box, choose the AddLink action and click the Go button. Following are the parameters present in AddLink window:

LinkName	The name of the link which is to be added. Default Value: SampleLink
SourceServerInitialContextFactory	The class name of the initial context factory which is used to open connection with the source server. Default Value: fiorano.jms.runtime.naming.FioranoInitialContextFactory
SourceServerName	The name of the source server. Default Value: SourceServer
SourceServerPassword	The password which the bridge uses to connect to the source server. Default Value: senna- The default password
SourceServerProtocol	The transport protocol used for establishing a connection with the source server. Default Value: TCP - The default transport protocol used for the connection.
SourceServerQCF	The connection factory with which the connection is to be established. Default Value: primaryQCF - The default connection factory used for the connection.
SourceServerSecurityCredentials	The SecurityCredentials (password) used for connecting to the

	source server. Default Value: Anonymous - The default password
SourceServerSecurityManager	ServerSecurityManager of the source server linked by the bridge. Default Value: null - No server Security Manager
SourceServerSecurityPrincipal	The SecurityPrincipal (username) used for connecting to the source server. Default Value: Anonymous- The default username
SourceServerType	Need to mention the type of the source server. Default Value: JMS - The default type of server
SourceServerURL	The URL of the source server should be given here. Default Value: http://localhost:1856 - The default provider URL of the server.
SourceServerUserName	The UserName which the bridge uses to connect to the source server. Default Value: ayrton- The default bridge user name
TargetServerInitialContextFactory	The class name of the initial context factory which is used to open connection with the target server. Default Value: fiorano.jms.runtime.naming.FioranoInitialContextFactory
TargetServerName	The name of the target server. Default Value: TargetServer
TargetServerPassword	The password which the bridge uses to connect to the target server. Default Value: senna- The default password
TargetServerProtocol	The transport protocol used for establishing a connection with the target server. Default Value: TCP - The default transport protocol used for the connection.
TargetServerQCF	The connection factory with which the connection is to be established. Default Value: primaryQCF - The default connection factory used for the connection.

TargetServerSecurityCredentials	The SecurityCredentials (password) used for connecting to the target server. Default Value: Anonymous - The default password
TargetServerSecurityManager	ServerSecurityManager of the target server linked by the bridge. Default Value: null - No server Security Manager
TargetServerSecurityPrincipal	The SecurityPrincipal (username) used for connecting to the target server. Default Value: Anonymous- The default username
TargetServerType	Need to mention the type of the target server. Default Value: JMS - The default type of server
TargetServerURL	The URL of the target server should be given here. Default Value: http://localhost:1856 - The default provider URL of the server.
TargetServerUserName	The UserName which the bridge uses to connect to the target server. Default Value: ayrton- The default bridge user name

To perform any of the actions listed in the Select Action list-box, choose the appropriate action and click the **Go** button.


FioranoMQ Web Manager - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://192.168.1.46:1781/#ConfigureBridge
Google
Fiorano
Web Management Console
User:admin
FioranoMQ 9.2.0, Build No. 9313, JVM Version : 32bit
Server Startup Time :Sat Mar 13 17:24:28 IST 2010
SaveConfigurations Shutdown Restart Logout
Users
Groups
StandAloneBridge
ConfigureBridge
DashBoard
Logs
JMX
AdvancedProperties
Licenses
SystemProperties
Configure StandAloneBridge
Select Action : Add Link Go
Fiorano Bridge
LocalHostToLocalHostFioranoBridge
FMQToFMQ-OneWay
FMQChannel1
AddLINK
Add
Cancel

LinkName	SampleLink
SourceServerInitialContextFactory	fiorano.jms.runtime.namin
SourceServerName	SourceServer
SourceServerPassword	senna
SourceServerProtocol	TCP
SourceServerQCF	primaryQCF
SourceServerSecurityCredentials	anonymous
SourceServerSecurityManager	null
SourceServerSecurityPrincipal	anonymous
SourceServerType	JMS
SourceServerURL	http://localhost:1856
SourceServerUserName	ayrton

Figure 32.26: AddLink

AddChannel Attributes

To add a channel to a link select Action list-box, choose the AddChannel action and click the Go button. Following are the parameters present in AddChannel window:

AddSourceQueueParams	Specify the source queue parameters. Default Value:undefined
AddTargetQueueParams	Specify the target queue parameters. Default Value:undefined
ChannelName	The name of the channel which is to be added. Default Value: channel1
SourceQueueMessageSel	The message selector of the queue. Allows the user to set message selectors for the source queues which have been linked using bridges. Default Value: null - No message selector is used
SourceQueueName	The name of the queue present in source server

	which is to be linked by the bridge. Default Value: primaryQueue - The name of the queue which the bridge links.
TargetQueueMessageSel	The message selector of the queue. Allows the user to set message selectors for the target queues which have been linked using bridges. Default Value: null - No message selector is used
TargetQueueName	The name of the queue present in target server which is to be linked by the bridge. Default Value: secondaryQueue - The name of the queue which the bridge links.

17. **StandAloneRepeater**

This tab is available when logged into the web console of a Server running on the StandAloneRepeater profile.

A Repeater transmits a message received on one topic (source topic) to another topic (target topic).

ConfigureRepeater view lists all the Repeaters currently available on the server. The Select Action list-box contains various options such as:

- View Repeater Attributes
- Show Repeater Info
- Add Link
- Delete Link
- View Link Configuration
- Edit Link Configuration
- Add Link Topic
- Show Link Topic Configurations
- Remove Link Topic
- Add Reply Topic
- Show Reply Topic Configurations
- Remove Reply Topic

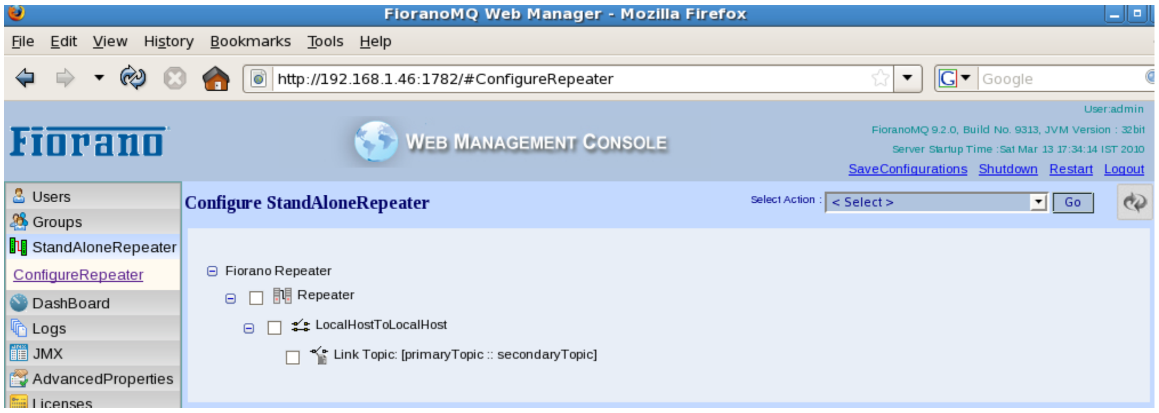
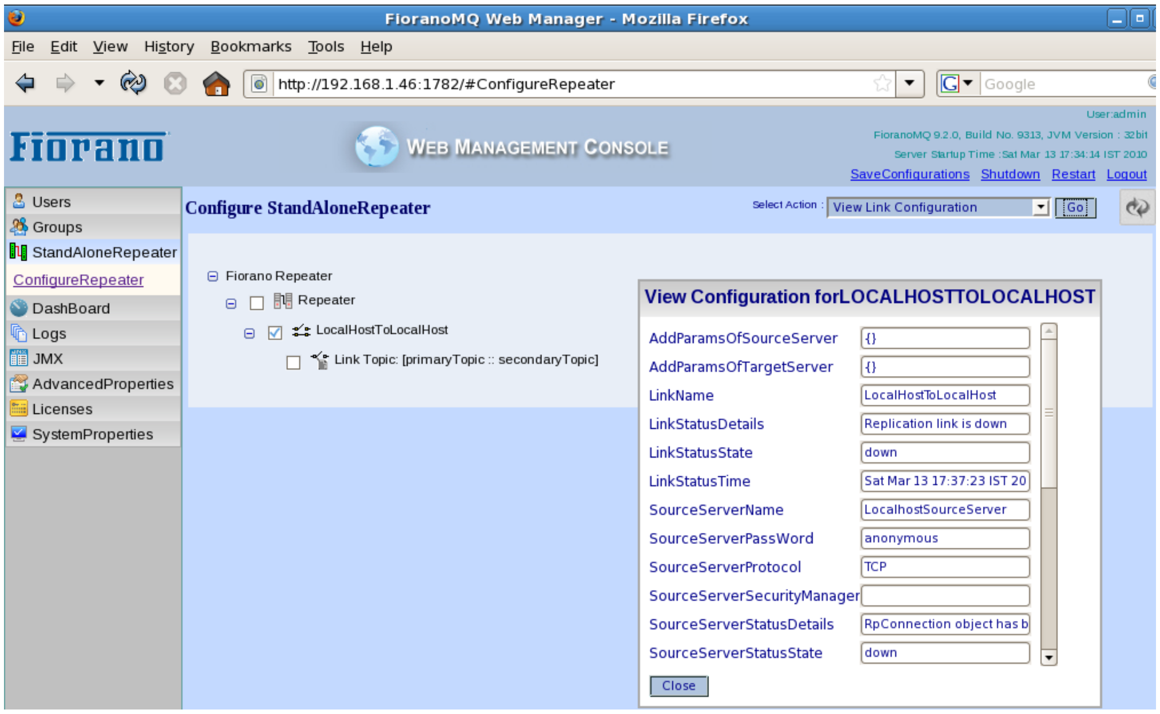

FioranoMQ Web Manager - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://192.168.1.46:1782/#ConfigureRepeater
Google
Fiorano
Web Management Console
User:admin
FioranoMQ 9.2.0, Build No. 9313, JVM Version : 32bit
Server Startup Time :Sat Mar 13 17:34:14 IST 2010
SaveConfigurations Shutdown Restart Logout
Users
Groups
StandAloneRepeater
ConfigureRepeater
DashBoard
Logs
JMX
AdvancedProperties
Licenses
Configure StandAloneRepeater
Select Action : < Select >
Go
Fiorano Repeater
Repeater
LocalHostToLocalHost
Link Topic: [primaryTopic :: secondaryTopic]

Figure 32.27: StandAloneRepeater

To perform any of the actions listed in the Select Action list-box, choose the appropriate action and click the **Go** button.


FioranoMQ Web Manager - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://192.168.1.46:1782/#ConfigureRepeater
Google
Fiorano
Web Management Console
User:admin
FioranoMQ 9.2.0, Build No. 9313, JVM Version : 32bit
Server Startup Time :Sat Mar 13 17:34:14 IST 2010
SaveConfigurations Shutdown Restart Logout
Users
Groups
StandAloneRepeater
ConfigureRepeater
DashBoard
Logs
JMX
AdvancedProperties
Licenses
SystemProperties
Configure StandAloneRepeater
Select Action : View Link Configuration
Go
Fiorano Repeater
Repeater
LocalHostToLocalHost
Link Topic: [primaryTopic :: secondaryTopic]
Close

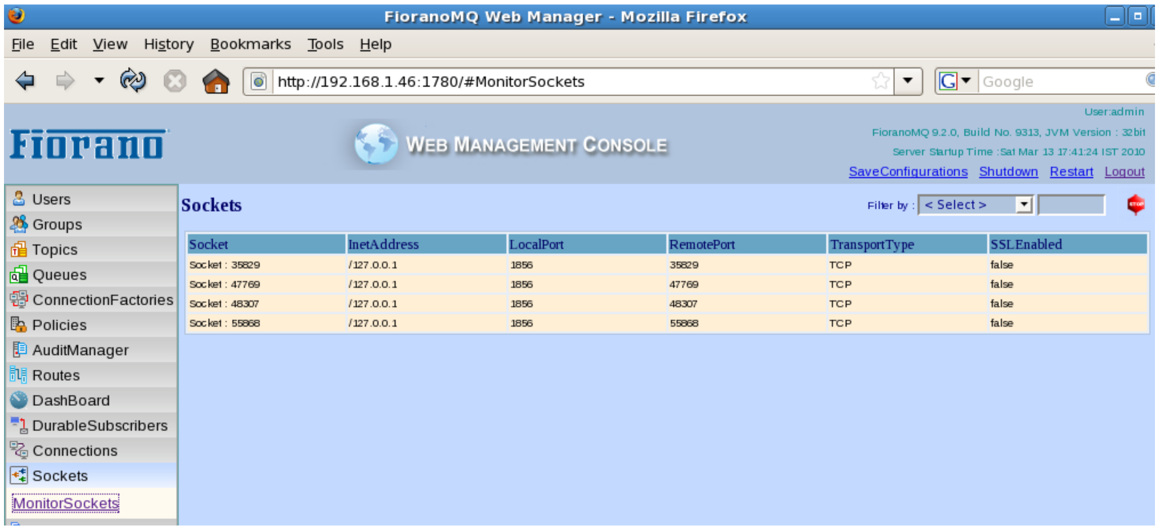
View Configuration forLOCALHOSTTOLOCALHOST

AddParamsOfSourceServer	{}
AddParamsOfTargetServer	{}
LinkName	LocalHostToLocalHost
LinkStatusDetails	Replication link is down
LinkStatusState	down
LinkStatusTime	Sat Mar 13 17:37:23 IST 20
SourceServerName	LocalhostSourceServer
SourceServerPassWord	anonymous
SourceServerProtocol	TCP
SourceServerSecurityManager	
SourceServerStatusDetails	RpConnection object has b
SourceServerStatusState	down

Figure 32.28: View Link Configuration

18. **Sockets**

MonitorSockets view contains information about all the sockets currently open within the Server. The sockets are created for each of the connections made to the server by the clients. The table gives information on Server ports, about the client, the transport type, IP address of the Server and whether SSL is enabled.



http://192.168.1.46:1780/#MonitorSockets

User:admin
FioranoMQ 9.2.0, Build No. 9313, JVM Version : 32bit
Server Startup Time :Sat Mar 13 17:41:24 IST 2010

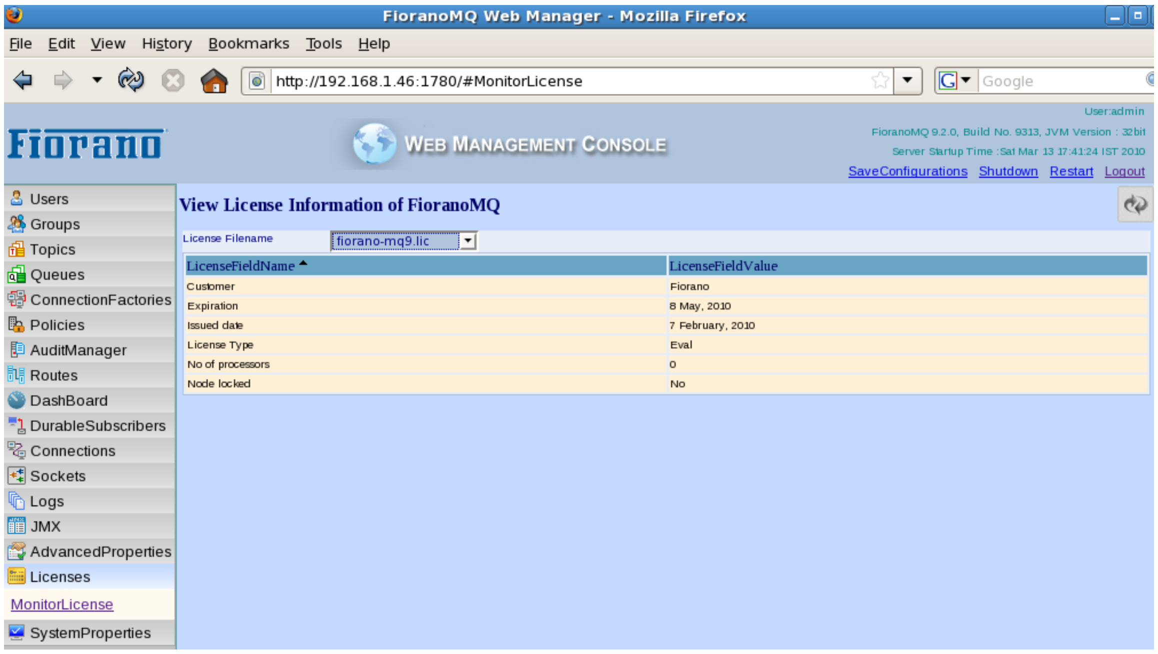
Sockets

Socket	InetAddress	LocalPort	RemotePort	TransportType	SSLEnabled
Socket : 35829	/127.0.0.1	1856	35829	TCP	false
Socket : 47769	/127.0.0.1	1856	47769	TCP	false
Socket : 48307	/127.0.0.1	1856	48307	TCP	false
Socket : 55868	/127.0.0.1	1856	55868	TCP	false

Figure 32.29: Monitor Sockets

19. **Licenses**

MonitorLicense displays information of about all the licenses available. The License Filename list-box lists the available licenses. Selecting a licensedisplays its corresponding license information in the table shown in the figure below:



http://192.168.1.46:1780/#MonitorLicense

User:admin
FioranoMQ 9.2.0, Build No. 9313, JVM Version : 32bit
Server Startup Time :Sat Mar 13 17:41:24 IST 2010

View License Information of FioranoMQ

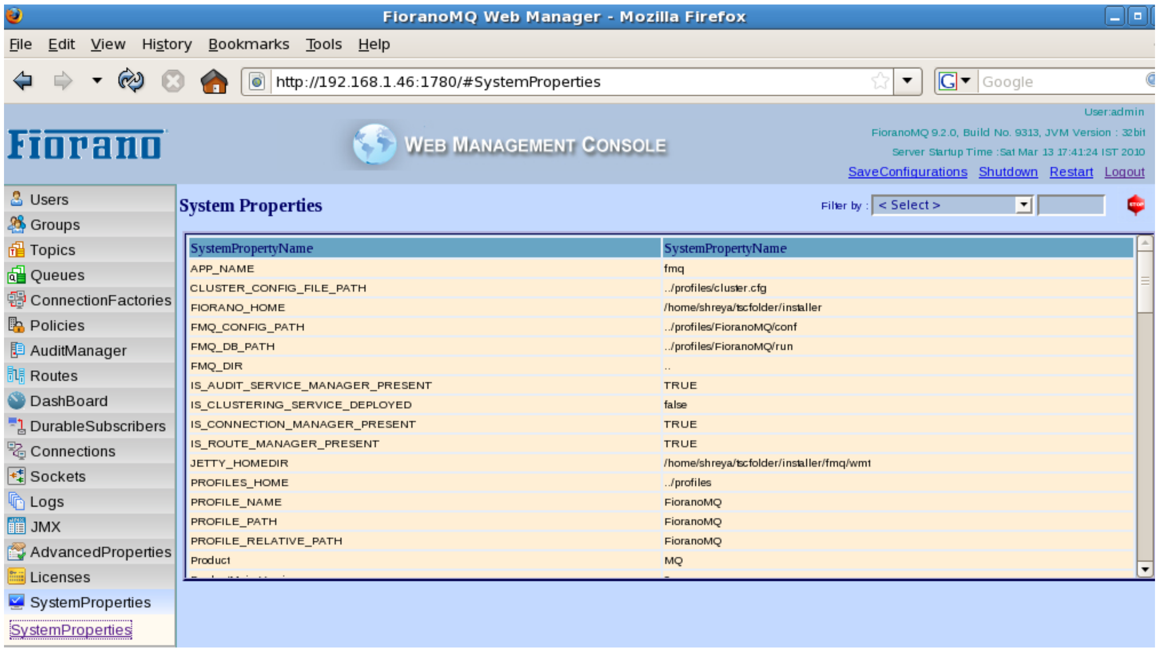
License Filename: fiorano-mq9.lic

LicenseFieldName	LicenseFieldValue
Customer	Fiorano
Expiration	8 May, 2010
Issued date	7 February, 2010
License Type	Eval
No of processors	0
Node locked	No

Figure 32.30: Licenses

20. **SystemProperties**

SystemProperties view lists the all the relevant properties of the system on which the FioranoMQ Server is hosted. These values cannot be modified through the WMT.



FioranoMQ 9.2.0, Build No. 9313, JVM Version : 32bit

Server Startup Time :Sat Mar 13 17:41:24 IST 2010

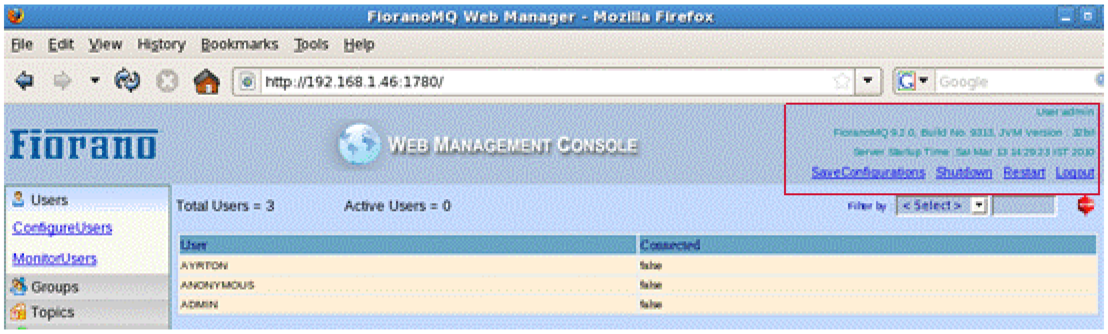
System Properties

SystemPropertyName	SystemPropertyName
APP_NAME	fmq
CLUSTER_CONFIG_FILE_PATH	../profiles/cluster.cfg
FIORANO_HOME	/home/shreya/tscfolder/installer
FMQ_CONFIG_PATH	../profiles/FioranoMQ/conf
FMQ_DB_PATH	../profiles/FioranoMQ/run
FMQ_DIR	..
IS_AUDIT_SERVICE_MANAGER_PRESENT	TRUE
IS_CLUSTERING_SERVICE_DEPLOYED	false
IS_CONNECTION_MANAGER_PRESENT	TRUE
IS_ROUTE_MANAGER_PRESENT	TRUE
JETTY_HOMEDIR	/home/shreya/tscfolder/installer/fmq/wmt
PROFILES_HOME	../profiles
PROFILE_NAME	FioranoMQ
PROFILE_PATH	FioranoMQ
PROFILE_RELATIVE_PATH	FioranoMQ
Product	MQ

Figure 32.31: System Properties

Status Information

The JVM version indicates whether the server JVM is 32 or 64 bits. This information, along with the information of when the FioranoMQ Server was started is displayed at the top right-hand corner of the WMT screen.



FioranoMQ 9.2.0, Build No. 9313, JVM Version : 32bit

Server Startup Time :Sat Mar 13 14:29:23 IST 2010

Total Users = 3 Active Users = 0

User	Connected
AYRTON	false
ANONYMOUS	false
ADMIN	false

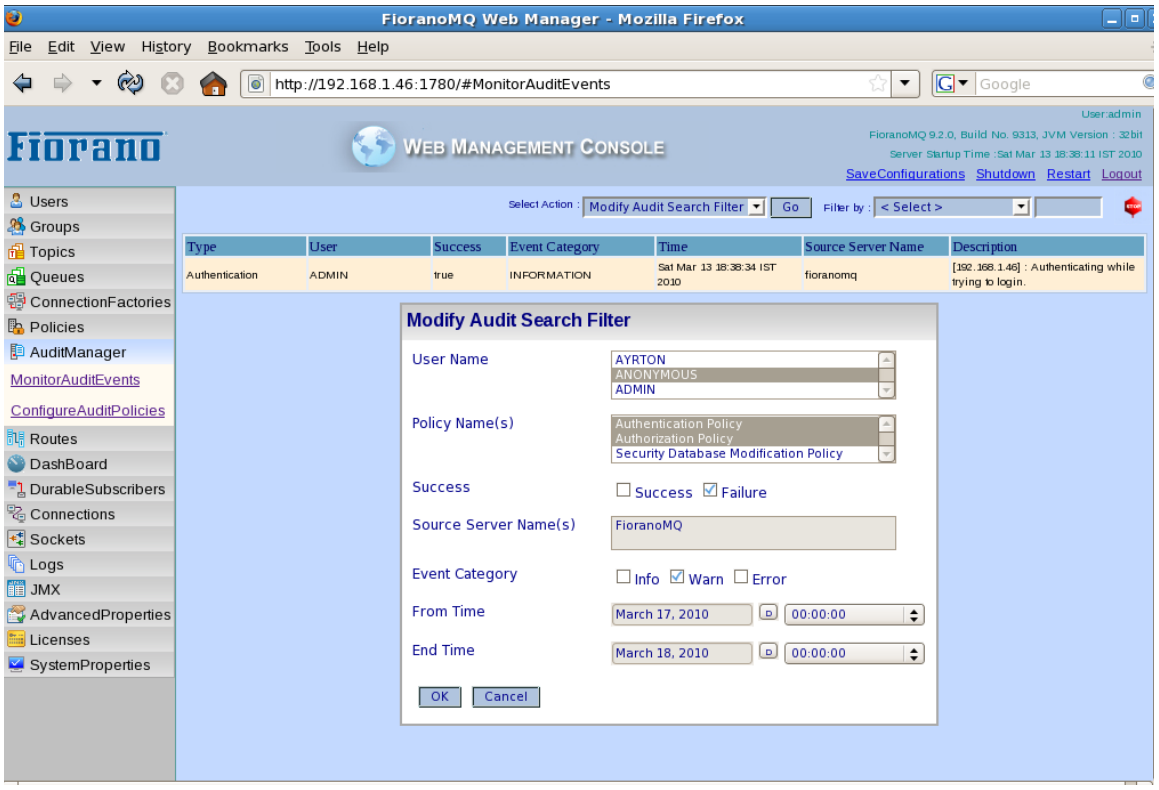
Figure 32.32: Status information

21. AUDIT MANAGER

Using the Audit Manager view it is possible to configure the Audit Storage Policies as well as monitor the audit events recorded in the Server.

Monitoring Audit Events

In the Monitoring Audit Events view, the audit events that are recorded in the System can be monitored. To enable this, the 'Audit Filter' set for the current view needs to be changed. The figure below shows how to modify the view:


http://192.168.1.46:1780/#MonitorAuditEvents

User:admin
FioranoMQ 9.2.0, Build No. 9313, JVM Version : 32bit
Server Startup Time :Sat Mar 13 18:38:11 IST 2010

Select Action : Modify Audit Search Filter

Type	User	Success	Event Category	Time	Source Server Name	Description
Authentication	ADMIN	true	INFORMATION	Sat Mar 13 18:38:34 IST 2010	fioranomq	[192.168.1.46] : Authenticating while trying to login.

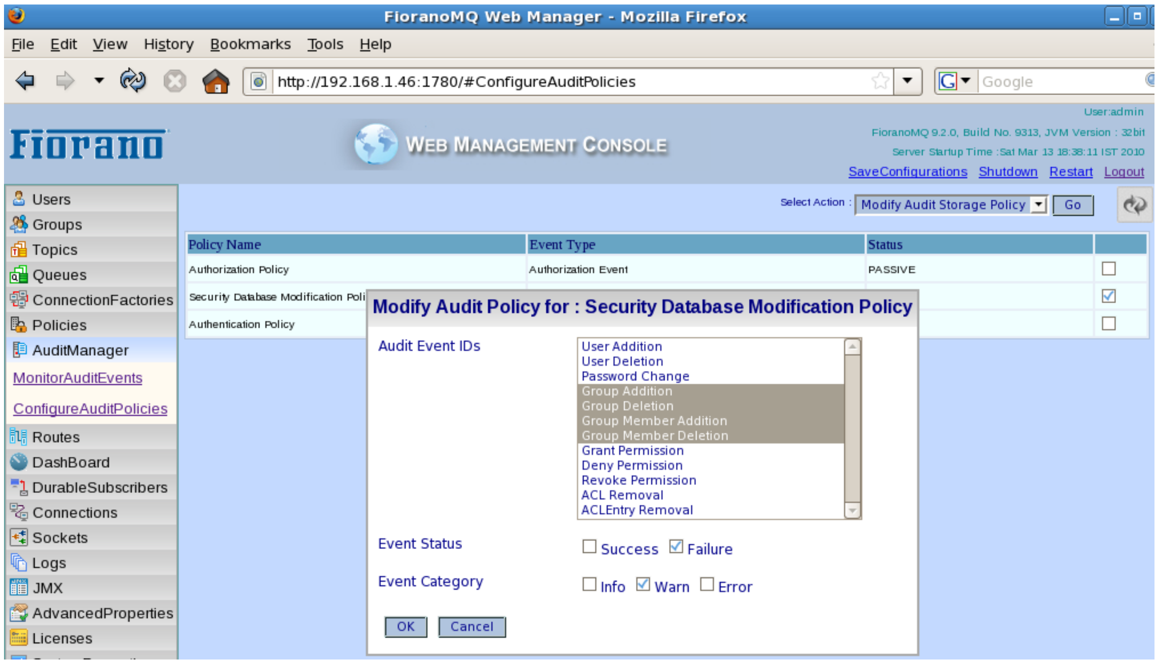
Modify Audit Search Filter

User Name	AYRTON, ANONYMOUS, ADMIN
Policy Name(s)	Authentication Policy, Authorization Policy, Security Database Modification Policy
Success	☐ Success ☑ Failure
Source Server Name(s)	FioranoMQ
Event Category	☐ Info ☑ Warn ☐ Error
From Time	March 17, 2010 00:00:00
End Time	March 18, 2010 00:00:00

Figure 32.33: Modify Audit Search Filter

Configuring Audit Policies

In the Configuring Audit Policies view, the various kinds of Storage Policies which are responsible for generating Audit events are configured. In order to do this, the action 'Modify Audit Storage Policy' needs to be invoked after selecting one of the Storage Policies listed in the table inside the view. The storage policy will then need to be configured. The figure below shows how to configure the storage policy:


FioranoMQ Web Manager - Mozilla Firefox

http://192.168.1.46:1780/#ConfigureAuditPolicies

User:admin
FioranoMQ 9.2.0, Build No. 9313, JVM Version : 32bit
Server Startup Time :Sat Mar 13 18:38:11 IST 2010

Select Action : Modify Audit Storage Policy

Policy Name	Event Type	Status	
Authorization Policy	Authorization Event	PASSIVE	☐
Security Database Modification Poli			☑
Authentication Policy			☐

Modify Audit Policy for : Security Database Modification Policy

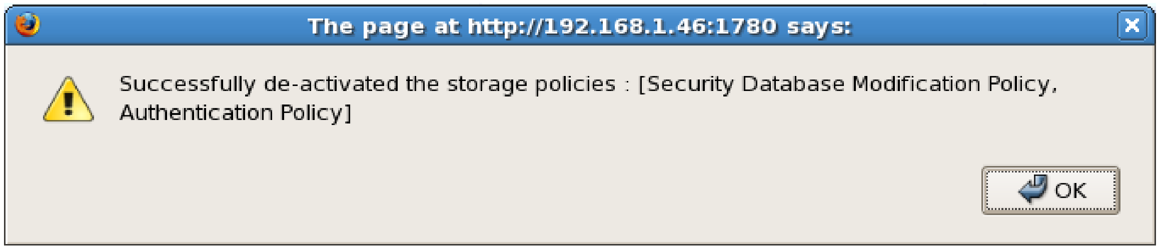
Audit Event IDs: User Addition, User Deletion, Password Change, Group Addition, Group Deletion, Group Member Addition, Group Member Deletion, Grant Permission, Deny Permission, Revoke Permission, ACL Removal, ACLEntry Removal

Event Status: ☐ Success ☑ Failure

Event Category: ☐ Info ☑ Warn ☐ Error

Figure 32.34: ConfigureAuditPolicies > Modify Audit Storage Policy

The recording of Audit events can be stopped by configuring the same storage policy that is used for enabling Audit Events. In order to do this, the action 'De-activate Storage Policy' needs to be invoked after selecting the Storage Policy which is to be de-activated. The figure below shows how to deactivate the storage policy:.


The page at http://192.168.1.46:1780 says:

Successfully de-activated the storage policies : [Security Database Modification Policy, Authentication Policy]

Figure 32.35: De-Activate Storage Policy message

For more information on Audit Management, please refer to Chapter 30 Audit Management of this document.

Chapter 33: Fiorano Directory Services

33.1 Introduction

In the real world application of messaging Servers, it is a common requirement to collectively manage a group of Servers from a remote place. 'Managing', entails the management of Server configurations. Every FioranoMQ Server requires a profile from which to boot up. This profile is in effect the configuration with which the Server starts. This Chapter explains the need for profile management of a cluster of messaging servers running at different geographical locations. And, this Chapter explains the solution that Fiorano provides to manage the profiles (the Server configurations) from a remote location.

33.1.1 Profile Management

FioranoMQ comes with a new component called **Fiorano Directory Services** (FDS) for managing Server configurations/profiles globally. The following subsections describe the use of the Fiorano Directory Services (FDS) and the corresponding profile management services. The default FioranoMQ installation comes with pre-created profiles that are configured to demonstrate certain Server functions. The default FioranoMQ profiles are local a particular instance of the FioranoMQ Server and changes made to that profile are applicable to that instance only.

Thus, when two or more FioranoMQ Servers (from different installations) start with the same profile, their configurations could very well be different (if the profiles have been modified in their respective installations). However, the profiles need to be consistent across several FioranoMQ Server instances. Therefore, the changes applied should be applied globally so as affected all instances. In earlier versions of FioranoMQ, changes made dynamically to the configurations of a Server running on a particular profile could not be propagated to other Servers running on the same profile (albeit on different installations). Once again, the only way to change the configuration (dynamically) of all Servers running on a particular profile would be to modify the configuration of each of the Servers individually. In order to start multiple instances of the FioranoMQ Servers with the same profile configurations there needs to be a central management component which also takes care of propagating any changes made to the messaging Servers in a cluster.

The FDS feature has been introduced to extend the profile configuration and management globally.This feature is used to support certain key enhancements in the current Clustering features. Every instance of FioranoMQ installation supports the profile management using the FDS component. It is therefore possible to run the FioranoMQ Server either with FDS deployed for profile management purposes or without FDS deployed, as a normal FioranoMQ Server for messaging purposes. FDS provides the ability to expose all the attributes of a profile and propagate the changes made to all the Servers using this profile. The following sections gives a detailed description of profile management and various operations such as adding a new customized profile, editing and deleting a profile from FioranoMQ Web Console and so on.

33.1.2 Components / Terminology

As discussed in the earlier section, the FioranoMQ Server has been modified to run as a 'Cluster Manager', which is also known as the 'FioranoMQ Management Server'. A FioranoMQ Server with the FDS component deployed and running is termed as a FioranoMQ Management Server.

The FDS component is deployed only when the server is run with a profile called 'FioranoMQ_ClusterManager'. However, the FioranoMQ Management Server can also be used for messaging purposes.

A FioranoMQ Server without the FDS component is termed as a FioranoMQ Messaging Server.

33.2 FDS Concepts

33.2.1 FioranoMQ Management Server

The FioranoMQ Management Server is a FioranoMQ Server which holds the added responsibilities of handling the profile (Server configurations) management globally. The FioranoMQ Management server interacts with the Apache Directory Services for communicating all the configuration parameters to other FioranoMQ messaging servers that are connected. The Directory Service manager handles all the interactions with the FioranoMQ Messaging Servers and translates them into requests understandable by the Apache Directory Service that store all the global profiles.

33.2.1.1 Cluster Manager

The FioranoMQ Management Server processes all the requests of its clients, i.e. other FioranoMQ messaging Servers, through the Cluster Manager component, i.e. the Directory Services Manager MBean. The FioranoMQ messaging Servers communicate with the FioranoMQ management Server through an RMI connection. The Cluster Manager is also responsible for transferring changes made to the Configs.cfg file or to the profile entry in DS to all the messaging servers using that profile. Each profile entry in the FDS will possess information about all the Servers that have booted using that profile.

33.2.1.2 How to run FioranoMQ Management Server

The FioranoMQ installer generates a set of preconfigured profiles within which the 'FioranoMQ_ClusterManager' profile is used for running the FioranoMQ Management Server.

To run in Windows or UNIX machines please navigate to $FIORANO_HOME/fmq/bin and execute startCluster.bat (.sh) which uses the 'FioranoMQ_ClusterManager' profile.

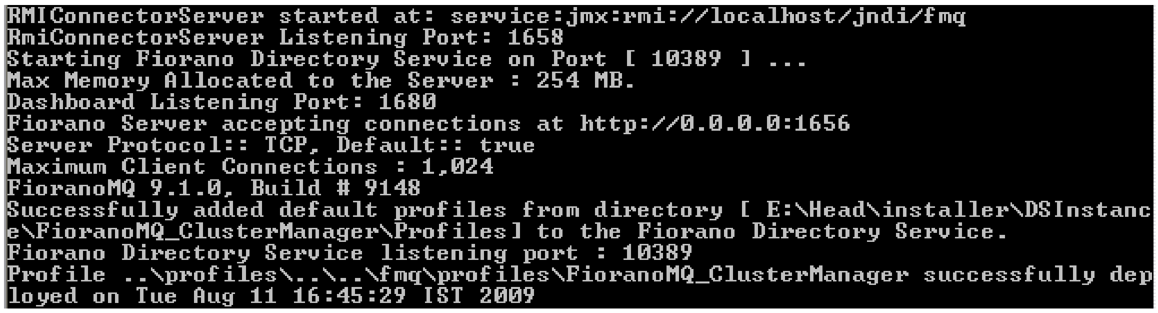
Below are the ports used for running the FioranoMQ Management Server. These ports are different from those used by a normal messaging Server.

- Accepting Connections at: 1656
- RMI Port: 1658
- Dashboard Listening Port: 1680

- Directory Services default Port: 10389

When the FioranoMQ Management server is run, it loads the FDS component and starts the Apache Directory Services on the default port (10389). When it is started for the first time all the default profiles are loaded into the Directory Services and areused only for global access and modifications.

The console of the FMQ Management Server when run will display :



```
RMIConnectorServer started at: service:jmx:rmi://localhost/jndi/fmq
RmiConnectorServer Listening Port: 1658
Starting Fiorano Directory Service on Port [ 10389 ] ...
Max Memory Allocated to the Server : 254 MB.
Dashboard Listening Port: 1680
Fiorano Server accepting connections at http://0.0.0.0:1656
Server Protocol:: TCP, Default:: true
Maximum Client Connections : 1,024
FioranoMQ 9.1.0, Build # 9148
Successfully added default profiles from directory [ E:\Head\installer\DSInstanc
e\FioranoMQ_ClusterManager\Profiles] to the Fiorano Directory Service.
Fiorano Directory Service listening port : 10389
Profile ..\profiles\..\..\fmq\profiles\FioranoMQ_ClusterManager successfully dep
loyed on Tue Aug 11 16:45:29 IST 2009
```

When run for the first time, a message, below, is displayed:

Successfully added default profiles from directory [E:\Fiorano\FioranoMQ9.1.0\DSInstance\FioranoMQ_ClusterManager\Profiles] to the Fiorano Directory Service.

Default profiles available at $FIORANO_HOME/fmq/profiles are loaded to the Apache Directory Services when the FDS is run for the first time and will persist as a separate instance under $FIORANO_HOME/DSInstance/ FioranoMQ_ClusterManager/profiles directory. Adding new profiles or modifications to the existing profiles will be effective only if done in this directory.

Configuring the DSInstance Path

The DSInstance path can be configured either using the Configs.cfg file of FioranoMQ_Clustering profile or using Studio in offline mode.

Given below are the steps to configure the DSInstance from the Studio:

1. Open the FioranoMQ_Clustering profile from the Studio in the offline mode.
2. Navigate to FioranoMQ_Clustering >> Fiorano >> etc >> DirectoryService >> InstancePath.
3. Set the InstancePath parameter to a valid absolute path.

Given below are the steps to configure DSInstance from Configs.cfg:

1. Open Configs.cfg file from the FioranoMQ_Clustering profile located at $FIORANO_HOME/fmq/profiles/FioranoMQ_Clustering/conf.
2. Under the **Clustering Manager Configuration Settings** group provide the absolute path for the parameter **InstancePath**.

The **InstancePath** parameter and theDirectory Services parameters are only available for the FioranoMQ_ClusterManager profile.

33.2.1.3 Propagation of modified attributes to FMQ Messaging Servers

As discussed previously, any configuration changes that are made to a profile that resides in the Directory Service will be propagated to all the FioranoMQ Messaging Servers which are registered under that particular profile. There may be some changes to the FioranoMQ parameters that require a Server restart to take effect. In this particular case, a check is done for the attributes that require restarting the Server and an appropriate notification is sent to the FioranoMQ Messaging Server.

Note:

- Local changes made to profiles of the individual Servers are not propagated and these changes affect only that particular instance of the messaging Server and not the profile in the central repository.
- Individual registered Servers decide whether they want to save global changes. If these changes are not saved, the local copy of the profile used by the Server will incorporate the modifications made to the profile located at the central repository.
- The changes made to the profile located at the central repository are propagated to all Servers running on the modified profile.

A notification to save the configurations and restart the server for the changes to get effective, is shown below. [Where? Is it written? Displayed?]

<Under what different conditions does the propagation of attributes not reach the messaging server?>

33.2.1.4 Registering & De-Registering Servers

The FioranoMQ Management Server is responsible for the maintenance of the server-profile information in Apache DS. When a FioranoMQ Messaging Server sends the notification to register itself under a profile, the Cluster manager adds an entry for the Server under the 'RegisteredServers' node of the profile entry in the DS. The Cluster Manager also adds an entry for the Server under the 'Servers' domain to keep track of the list of all registered servers present in the network. When the Cluster Manager receives a notification to remove the registration of a Server from a profile, it deletes the Server's entry from the 'RegisteredServers' node and from the 'Servers' domain.

33.2.1.5 Handling Network Failure

At the time of registering the FioranoMQ Messaging Server in the FioranoMQ Management Server, the Management Server continuously fetches the health status of the messaging server as per the configured ping intervals. If this request to fetch the health status (HEALTH_REQUEST_MAN) fails (as explained below), the corresponding messaging Server will be de-registered from the Management Server.

It is important to understand the two different parameters used while 'Pinging' the Messaging Servers:

- PingInterval: Time duration in Milliseconds for which the Pinging Thread will wait before sending the next request.

- PingTimeout: Timeout for each of the HEALTH_REQUEST_MANs after which an exception occurs and the corresponding messaging Server is de-registered.

These parameters are configurable in the Management Server's profile.

HEALTH_REQUEST_MAN may fail due to the reasons below:

- If Messaging Server is shutdown in a manner that is not acceptable.
- If Messaging Server's process exits abruptly.
- If Messaging Server is Out of Network.
- If Management Server is Out of Network.
- If Request takes an unusually long time and timeout occurs.

33.2.2 FMQ Messaging Server

The FioranoMQ Messaging Server acts as a normal MQ server and is configurable to connect to the MQ management Server. The FMQ messaging server makes an RMI connection with FioranoMQ Management Server to download a specified profile from which boot up. The messaging Servers get themselves registered under the profile entry of the profile that they are using that is available in the DS. The FioranoMQ Messaging Server gets notified of changes made to the profile in the FDS while it is running. Whena messaging Server shuts down, a notification is sent to remove its registration from the profile on which it was running.In instances of network failures a notification to remove the registration from the profile cannot be sent.

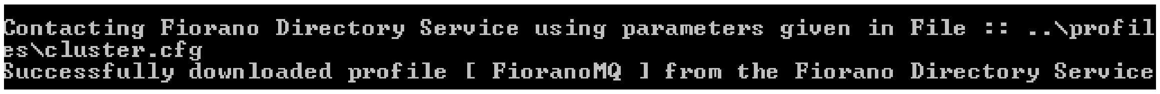
33.2.2.1 How to run FioranoMQ Messaging Server

A FioranoMQ Messaging server can be started with any of the desired profiles present in the FDS repository. The User is provided with an option to fetch the desired profile that is used for booting up. For instance, if the desired profile is <FioranoMQ_Machine1>, the FioranoMQ messaging Server should be started using:

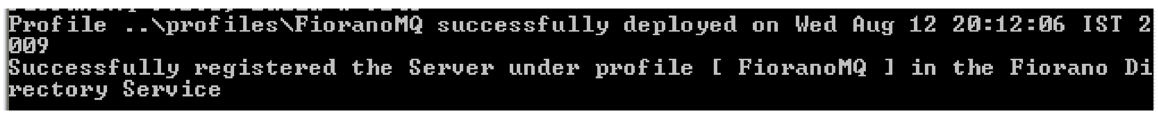
fmq.bat(.sh) –profile <FioranoMQ_Machine1>

If the desired profile exists in the FDS repository, it will be downloaded to this particular messaging Server's profiles' directory. If the desired profile does not exist, the messaging Server will try to boot up using a local profile with the same name. If this profile, too, is not present in the local directory, the Server will not boot up and will exit.

The console of the FioranoMQ messaging Server displays the information below when connected to FioranoMQ Management in order to fetch a profile and boot up using the same profile:


```
Contacting Fiorano Directory Service using parameters given in File :: ..\profil
es\cluster.cfg
Successfully downloaded profile [ FioranoMQ ] from the Fiorano Directory Service
```

The console should indicatethat it has successfully registered under the same profile with the FDS as shown below:



```
Profile ..\profiles\FioranoMQ successfully deployed on Wed Aug 12 20:12:06 IST 2
009
Successfully registered the Server under profile [ FioranoMQ ] in the Fiorano Di
rectory Service
```

Default behavior:

The behavior of the FioranoMQ Messaging Server is based on the properties cluster.cfg file located at $FIORANO_HOME/fmq/profiles. By default, the FioranoMQ Messaging Server is not configured to fetch or download profiles from the FMQ Management Server. Therefore, the properties listed below, within cluster.cfg file, need to be modified.

FETCH_PROFILE: This a Boolean value determining whether to fetch profile from the Management Server. By default this value is set to 'FALSE'.Because this value is set to 'FALSE' the FMQ Messaging Server doesnot connect to the FioranoMQ Management Server to fetch the desired profile and run as a normal FioranoMQ server

FETCH_PROFILE_ONLY_ONCE: This is a Boolean value determining whether to fetch a given profile when a FioranoMQ messaging Server is restarted. If set to 'TRUE', the Messaging Server which using the profile will not fetch the profile from the FMQ Management Server when restarted. In such instances the FioranoMQ Messaging Server will still be registered within the FioranoMQ Management Server. The default value this value is set to 'TRUE'.

This flag is valid only when the flag FETCH_PROFILE is set to true.

Configurable Parameters Within cluster.cfg

Below are the configurable parameters that can change the behavior of the FioranoMQ Messaging Server:

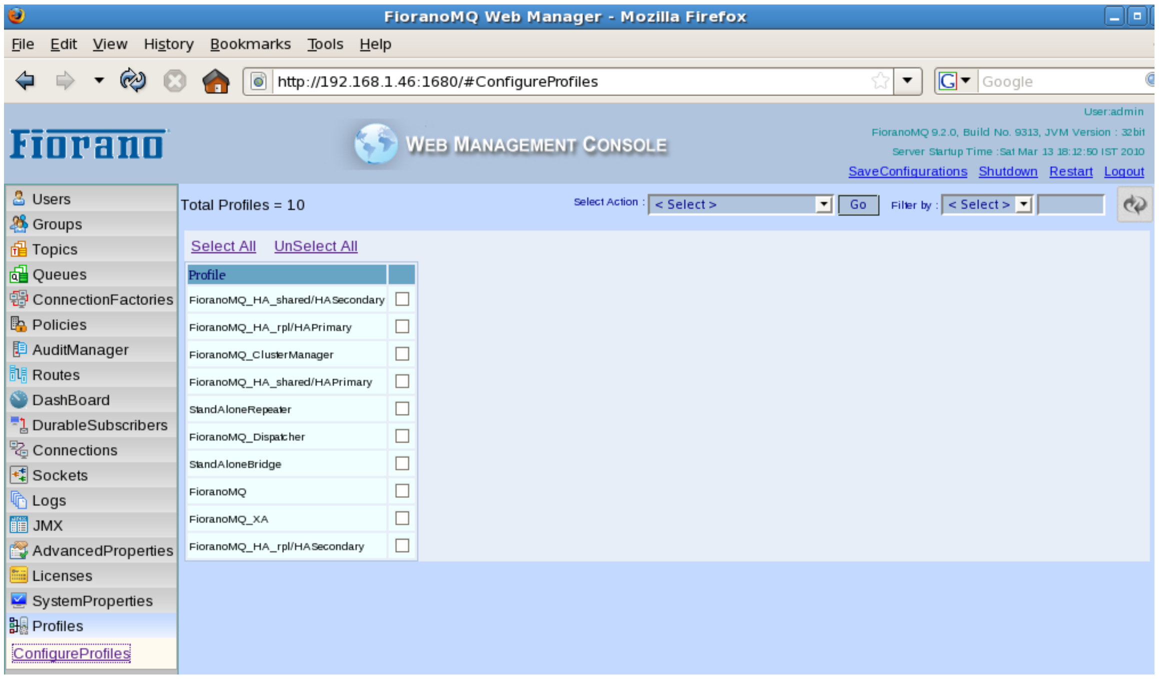
- **BIND_NAME=fmq** Bind name of the Management Server. This parameter need not be changed for normal users.
- **PROTOCOL=rmi** Protocol which will be used for making a connection to the Management Server's JMX server.
- **SERVER_ADDRESS=localhost** IPAddress of the Management Server where the JMX Service is started.
- **PORT=1658** RMI port which will be used for making a JMX Connection.
- **USERNAME=admin** User name which will be used for authentication while making a JMX Connection.
- **PASSWORD=passwd** Password which will be used for authentication while making a JMX Connection.
- **WAIT_BETWEEN_RECONNECT_ATTEMPTS=5000** This is the time, in milliseconds, the Messaging Server waits before trying to re-establish a connection with the Management Server.
- **PING_INTERVAL=20000** This is the time, in milliseconds, the Messaging Server waits before trying to check the health status of the Management Server.
- **PING_TIMEOUT=10000** This is the timeout duration, in milliseconds, for each ping request to check health status of the Management Server.

Please note that one should wait for the FioranoMQ Management Server to be running for the FioranoMQ Messaging Server to be successfully registered.

33.3 Managing Profiles Using Web Console

The FioranoMQ Web console replicates the profile structure of the Fiorano Directory Services and enables the administrator to make changes to an existing profile or add/delete profiles. The Directory Services tab in the Web Console provides a list of profiles and their attributes that exist in the DS that is embedded in the FioranoMQ Management Server.

Note: Only the FioranoMQ Management Server Web console hasthe 'profiles' tab. The Web console of the FioranoMQ Messaging Server does not have the 'profiles' tab available.


FioranoMQ Web Manager - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://192.168.1.46:1680/#ConfigureProfiles
Google
Fiorano
WEB MANAGEMENT CONSOLE
User:admin
FioranoMQ 9.2.0, Build No. 9313, JVM Version : 32bit
Server Startup Time :Sat Mar 13 18:12:50 IST 2010
SaveConfigurations Shutdown Restart Logout
Users
Groups
Topics
Queues
ConnectionFactories
Policies
AuditManager
Routes
DashBoard
DurableSubscribers
Connections
Sockets
Logs
JMX
AdvancedProperties
Licenses
SystemProperties
Profiles
ConfigureProfiles
Total Profiles = 10
Select Action : < Select > Go
Filter by : < Select >
Select All UnSelect All

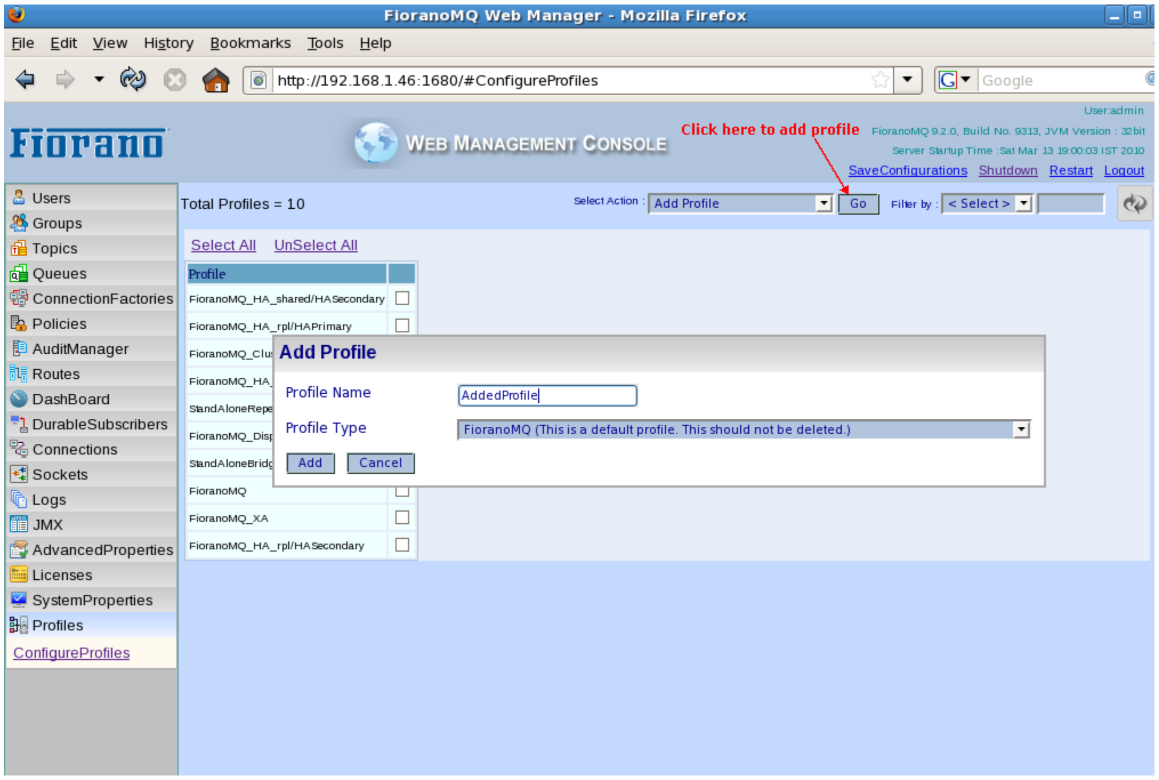
Profile	
FioranoMQ_HA_shared/HASecondary	☐
FioranoMQ_HA_rpl/HAPrimary	☐
FioranoMQ_ClusterManager	☐
FioranoMQ_HA_shared/HAPrimary	☐
StandAloneRepeater	☐
FioranoMQ_Dispatcher	☐
StandAloneBridge	☐
FioranoMQ	☐
FioranoMQ_XA	☐
FioranoMQ_HA_rpl/HASecondary	☐

33.3.1 Operations Performed using the web console

All Users who belong to 'Administrators' group can perform various operations such as Upload profile, Add profile, Delete profile, Edit profile, GetRegistered Servers, and Get All Registered Servers.

33.3.1.1 Adding Profile

To adding a new profile to FDS, a dialog box requesting inputs for the name of the profile and the profile 'type' is displayed. The profile type is one of the default profiles provided to customers upon which they can build their own profile with few changes to the Configs.cfg file parameters. The two screen-shots below display the Web console for the Add Profile function:



http://192.168.1.46:1680/#ConfigureProfiles

User:admin
FioranoMQ 9.2.0, Build No. 9313, JVM Version : 32bit
Server Startup Time :Sat Mar 13 19:00:03 IST 2010

Total Profiles = 10

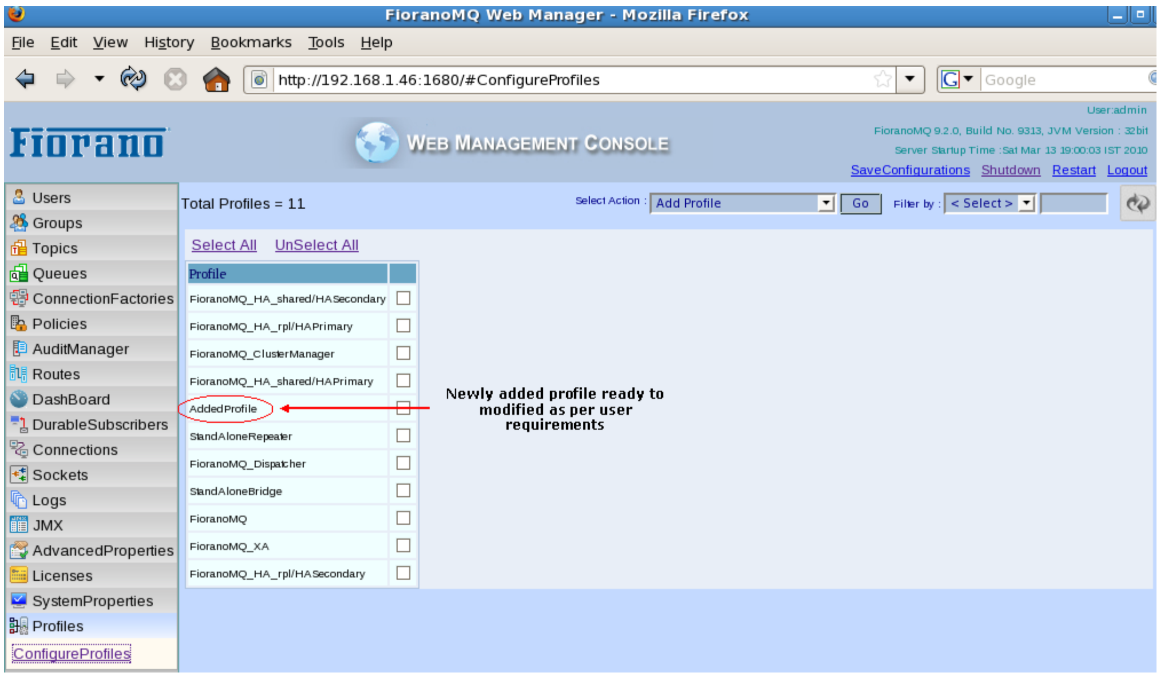
Select Action : Add Profile Go Filter by : < Select >

Profile	
FioranoMQ_HA_shared/HASecondary	☐
FioranoMQ_HA_rpl/HAPrimary	☐
FioranoMQ_Clu	
FioranoMQ_HA_	
StandAloneRepe	
FioranoMQ_Disp	
StandAloneBridg	
FioranoMQ	☐
FioranoMQ_XA	☐
FioranoMQ_HA_rpl/HASecondary	☐

Add Profile

Profile Name	AddedProfile
Profile Type	FioranoMQ (This is a default profile. This should not be deleted.)

On adding the profile ‘AddedProfile’, a copy of the profile mentioned in the **Profile Type** list box is created and renamed as name specified in the **Profile Name** text box. The Configs.cfg file of the newly created profile is parsed and all the necessary Mbeans and their corresponding attributes are loaded into the DS under the profile name ‘AddedProfile’. On successful addition of the profile, a message indicating that the addition of the profile has been successful is displayed on screen. The User can edit the profile added according to requirements as shown below:



http://192.168.1.46:1680/#ConfigureProfiles

User:admin
FioranoMQ 9.2.0, Build No. 9313, JVM Version : 32bit
Server Startup Time :Sat Mar 13 19:00:03 IST 2010

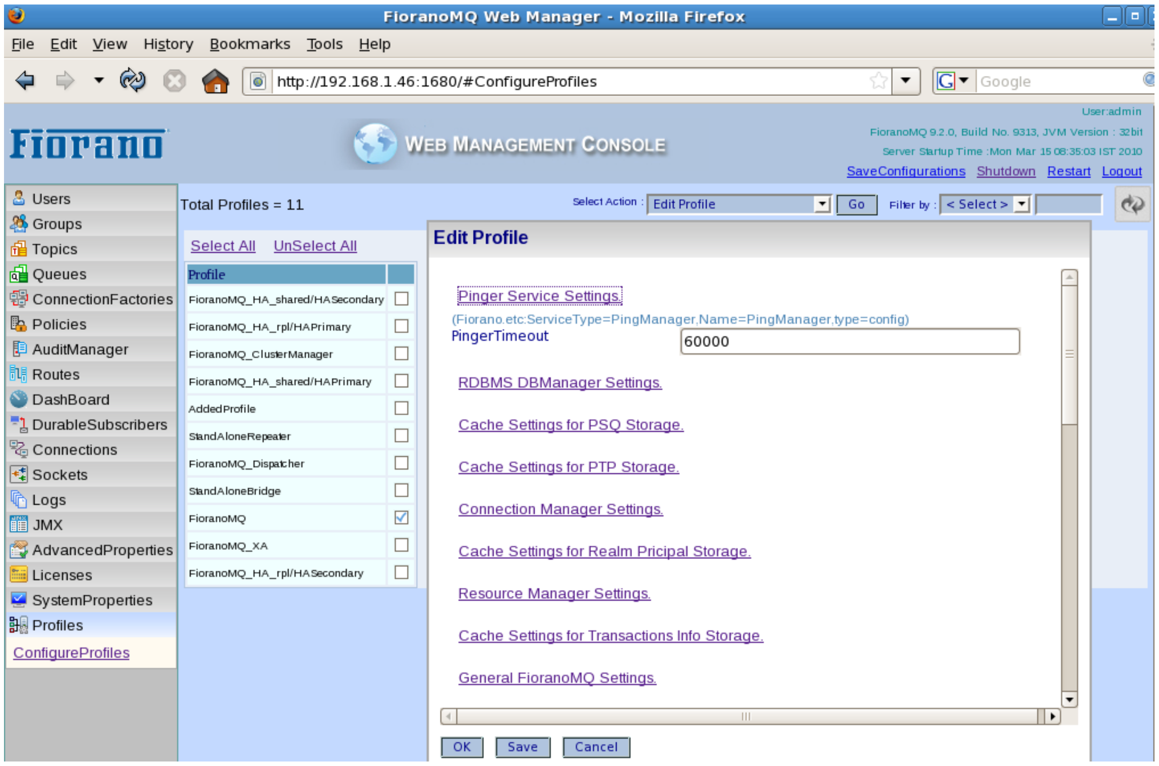
Total Profiles = 11

Select Action : Add Profile Go Filter by : < Select >

Profile	
FioranoMQ_HA_shared/HASecondary	☐
FioranoMQ_HA_rpl/HAPrimary	☐
FioranoMQ_ClusterManager	☐
FioranoMQ_HA_shared/HAPrimary	☐
AddedProfile	☐
StandAloneRepeater	☐
FioranoMQ_Dispatcher	☐
StandAloneBridge	☐
FioranoMQ	☐
FioranoMQ_XA	☐
FioranoMQ_HA_rpl/HASecondary	☐

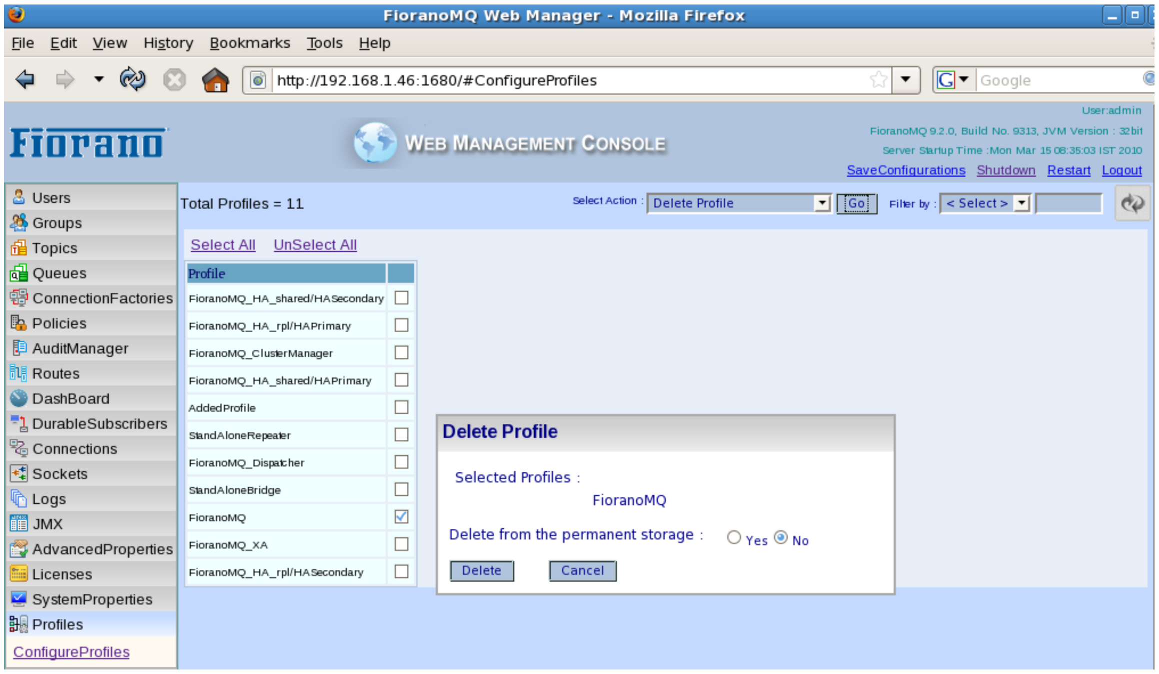
33.3.1.2 Editing Profile

The administrator can modify an existing profile via the edit option. The edit option dialog box opens showing the contents of the Configs.cfg file of the selected profile. All the Mbeans and their corresponding attributes with their values are displayed in the dialog box as a MBean tree structure. These values can be modified here. The changes made are reflected both in the Configs.cfg file of the profile present in the central repository and also in the virtual representation of the profile in the DS. These changes are then propagated to all the Servers (if any) currently running on the modified profile. The below screen-shot displays a preview of this:


FioranoMQ Web Manager - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://192.168.1.46:1680/#ConfigureProfiles
Google
Fiorano
Web Management Console
User:admin
FioranoMQ 9.2.0, Build No. 9313, JVM Version : 32bit
Server Startup Time :Mon Mar 15 08:35:03 IST 2010
SaveConfigurations Shutdown Restart Logout
Users
Groups
Topics
Queues
ConnectionFactories
Policies
AuditManager
Routes
DashBoard
DurableSubscribers
Connections
Sockets
Logs
JMX
AdvancedProperties
Licenses
SystemProperties
Profiles
ConfigureProfiles
Total Profiles = 11
Select Action : Edit Profile
Go
Filter by : < Select >
Select All UnSelect All
Profile
FioranoMQ_HA_shared/HASecondary
FioranoMQ_HA_rpl/HAPrimary
FioranoMQ_ClusterManager
FioranoMQ_HA_shared/HAPrimary
AddedProfile
StandAloneRepeater
FioranoMQ_Dispatcher
StandAloneBridge
FioranoMQ ✓
FioranoMQ_XA
FioranoMQ_HA_rpl/HASecondary
Edit Profile
Pinger Service Settings
(Fiorano.etc:ServiceType=PingManager,Name=PingManager,type=config)
PingerTimeout 60000
RDBMS DBManager Settings
Cache Settings for PSQ Storage
Cache Settings for PTP Storage
Connection Manager Settings
Cache Settings for Realm Pricipal Storage
Resource Manager Settings
Cache Settings for Transactions Info Storage
General FioranoMQ Settings
OK
Save
Cancel

33.3.1.3 Deleting Profile

Select all the profiles that need to be deleted from the DS and select the Delete Profile option from the Select Action list box. Click on 'Go'. A confirmation dialog box and a dialog box asking the User whether the profile needs to be delete from the disk is displayed. Upon confirmation the profiles are deleted (along with their Mbeans & Attribute Entries). However, if the profile to be deleted has Servers running on it or if it is the default profile, then deletion of the profile results in a failure and a message indicating the operation appears on the screen.


FioranoMQ Web Manager - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://192.168.1.46:1680/#ConfigureProfiles
Fiorano
Web Management Console
User:admin
FioranoMQ 9.2.0, Build No. 9313, JVM Version : 32bit
Server Startup Time :Mon Mar 15 08:35:03 IST 2010
SaveConfigurations Shutdown Restart Logout
Users
Groups
Topics
Queues
ConnectionFactories
Policies
AuditManager
Routes
DashBoard
DurableSubscribers
Connections
Sockets
Logs
JMX
AdvancedProperties
Licenses
SystemProperties
Profiles
ConfigureProfiles
Select All UnSelect All
Delete Cancel

Total Profiles = 11

Select Action : Delete Profile Go Filter by : < Select >

Profile	
FioranoMQ_HA_shared/HASecondary	☐
FioranoMQ_HA_rpl/HAPrimary	☐
FioranoMQ_ClusterManager	☐
FioranoMQ_HA_shared/HAPrimary	☐
AddedProfile	☐
StandAloneRepeater	☐
FioranoMQ_Dispatcher	☐
StandAloneBridge	☐
FioranoMQ	✓
FioranoMQ_XA	☐
FioranoMQ_HA_rpl/HASecondary	☐

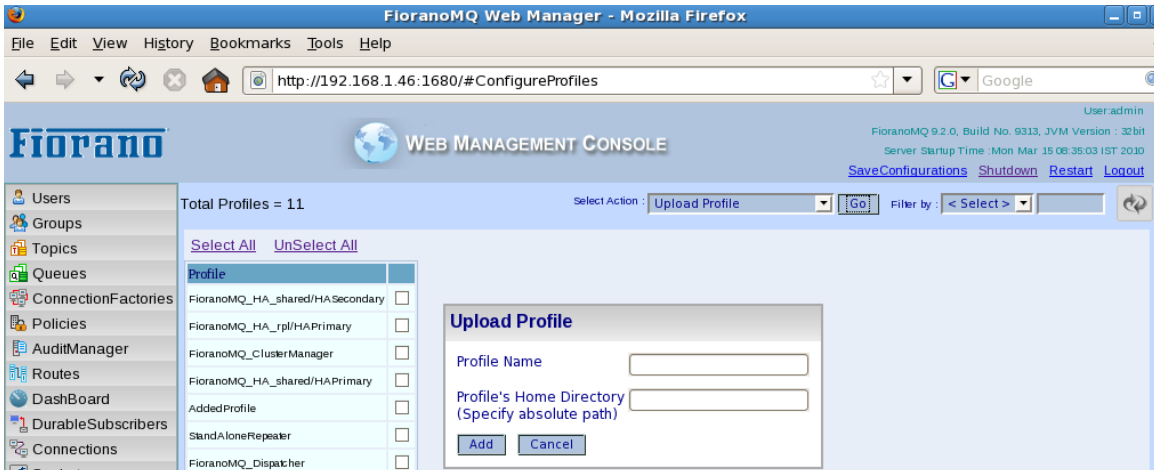
Delete Profile

Selected Profiles :
FioranoMQ

Delete from the permanent storage : Yes [✓ No]

33.3.1.4 Uploading Profile

The ‘Upload Profile’ option is used when a profile is present on the central repository but has not yet been added to the DS. On selecting the Upload Profile option, a dialog box is displayed asking the User to enter the profile's name and the path to its home directory. It then copies the profile into the %PROFILES_HOME% directory (if it is present in the %PROFILES_HOME% directory the copy operation is skipped) and adds a new entry into the DS for the profile. All of its Mbeans and attributes are then added as children of the profile entry by parsing the “Configs.cfg” of the added profile.


FioranoMQ Web Manager - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://192.168.1.46:1680/#ConfigureProfiles
Fiorano
Web Management Console
User:admin
FioranoMQ 9.2.0, Build No. 9313, JVM Version : 32bit
Server Startup Time :Mon Mar 15 08:35:03 IST 2010
SaveConfigurations Shutdown Restart Logout
Users
Groups
Topics
Queues
ConnectionFactories
Policies
AuditManager
Routes
DashBoard
DurableSubscribers
Connections
Select All UnSelect All
Add Cancel

Total Profiles = 11

Select Action : Upload Profile Go Filter by : < Select >

Profile	
FioranoMQ_HA_shared/HASecondary	☐
FioranoMQ_HA_rpl/HAPrimary	☐
FioranoMQ_ClusterManager	☐
FioranoMQ_HA_shared/HAPrimary	☐
AddedProfile	☐
StandAloneRepeater	☐
FioranoMQ_Dispatcher	☐

Upload Profile

Profile Name

Profile's Home Directory (Specify absolute path)

33.3.1.5 Get Registered Servers

Displays a list of IP address:port combination of the FioranoMQ Messaging Serversrunning under the selected profile.

33.3.1.6 Get All Registered Servers

Displays a list of IP address:port combination of the FioranoMQ Messaging Serversrunning under each profile.

33.4 Troubleshooting

1. Different kinds of logs related to the FioranoMQ Management Server and the FioranoMQ Messaging Server are:

   Logs related to the FioranoMQ Management Server:

   cluster.log: It is created in the dbPath of the FioranoMQ Management Server. All the logs for each operation on the Apache Directory Service are redirected to this log file.

   profile-monitor.log: All the propagational changes on 'profile edit' are directed to this log.

   Log related to the FioranoMQ Messaging Server:

   dsbroker.log: Logs for DS related operations are directed to this log. This log is present in PROFILES_HOME.

2. If the apache directory services do not start along with the FioranoMQ Management Server then to debug:

   All the error messages are redirected to cluster.log.

3. Different kind of exceptions in DS logs and their meaning:

   Different kinds Exceptions in DS logs are:

   a. "fiorano.jms.common.FioranoException: Error :: Profile is not found. :: Failed to find the profile"

      Indicates that the Server is trying to fetch a non-existent profile.

   b. "java.io.IOException: Failed to retrieve RMIServer stub"

      Indicates a network problem or that the management Server is not running when a profile is requested.

   c. "java.lang.SecurityException: Authentication Failed :: null"

      Occurs when the wrong username/password combinations for a RMI connection to the management Server is provided.

   d. "fiorano.jms.common.UnsuccessfulOperationException"

      Indicates that some operation has failed.

33.5 FAQ's

Question: What is profile Management?

Answer: Profile Management constitutes collectively managing profiles of a group of Servers from a remote location.

Question: What is Fiorano Directory Services (FDS)?

Answer: FDS is a profile management component of the FioranoMQ, which is used for managing profiles and/or Server configurations globally. FDS provides the ability to expose all the attributes of a profile and propagate the changes made to all Serversusing this profile.

Example:

When two or more FioranoMQ Servers (from different installations) start with the same profile, their configurations could very well be different (if the profiles have been modified in their respective installations). However, if these profiles need to be consistent across several FioranoMQ Server instances the changes should be done globally to affected each instance. FDS can be used to edit a profile globally for more than one FioranoMQ Servers and these changes are then reflected in both Servers.

Question: What is the difference between the FioranoMQ Messaging Server and FioranoMQ Management Server?

Answer: The FioranoMQ Messaging Server with the FDS component is termed as the FioranoMQ Management Server. The Management Server is used to manage the Messaging Server registered under it. The Management Server can also be used as a Messaging Server.

Question: How is the Management Server Run??

Answer: Please refer to section 33.2.2.1 How to run FioranoMQ Messaging Server.

Question: Can a New Profile be Uploaded to the Directory Services?

Answer: Yes, one can upload a new customized profile to Directory Services. For more information on how to upload please refer to section 33.3.1.4 Uploading Profile.

Question: How is a Profile in the Directory Service Deleted?

Answer: A profile that is being used by a messaging Server registered under the management Server or a default profile may not be deleted.

Question: What Operations can be Performed on Profiles in the Directory Services?

Answer: The operations that an administrator can perform on profiles in the Directory Services are:

- Upload profile
- Add profile
- Delete profile
- Edit profile
- Get registered server
- Get all registered server

For more information on these operations please refer to section 1.5.

Question: What Default Ports are used by the FioranoMQ_clusterManager profile and the FioranoMQ Management Server?

Answer: Default Ports used by FioranoMQ_clusterManager profile and the FioranoMQ Management Server are:

- Accepting Connections : 1656
- RMI Port : 1658
- Dashboard Listening Port: 1680
- Directory Services default Port: 10389

Chapter 34: Audit Management

Auditing provides an insight into the running of FioranoMQ while, more importantly, providing a means of accountability for changes administered within the system. Auditing is helpful in detecting security violations and conducting post-mortem analysis on information provided so as to find the root cause of a problem.

34.1 Audit Events

An audit event is a specific condition of the server or a particular change made in the server that is recorded. These events might range from the trivial to the vital. A policy is used to define the level of events that need be audited. Events might also be categorized based on functions such as Authentication, Authorization and Security Database Modification, where:

- **Authentication** - Covers all authentication requests within the MQ server.
- **Authorization** - Cover all authorization checks irrespective of whether a User is authorized to perform the action.
- **Security Database Modification** – Covers all changes made to the security store such as creation/deletion of User/Group, setting of ACL permissions and so on.

34.2 Audit Policies

The Server decides whether to store an audit event, based on the policies defined by the User. These policies (or Filters or Storage Policies or Audit Policies) are rules which define certain conditions. The server records all audit events that match these conditions. Policies are configuration tools that the User has at his disposal to fine tune the detail of the events to be recorded.

When an event is recorded, the server checks with its policies to see if the event need be recorded.

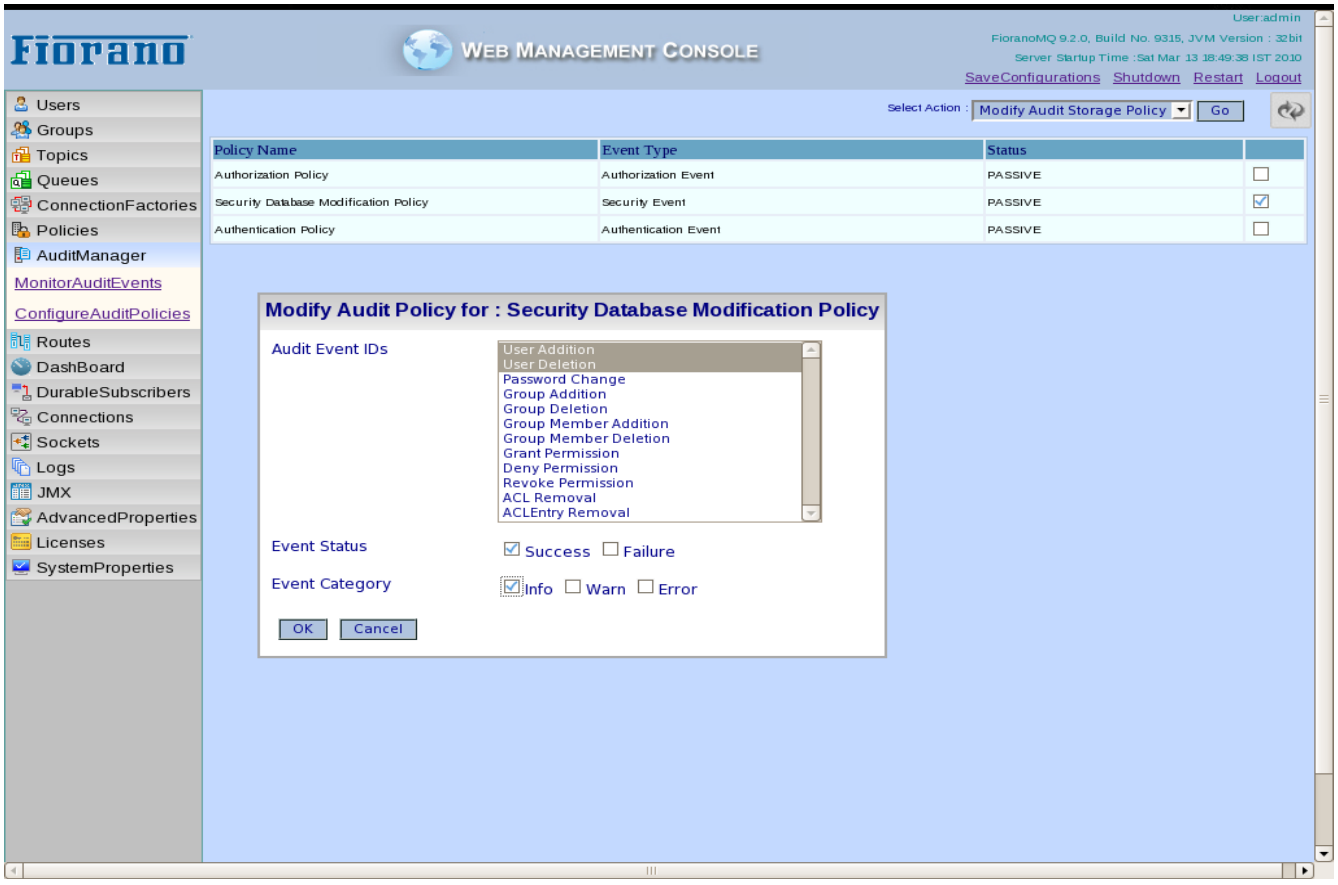
Note: So as to perform the actions specified in sections, Enabling Auditing and Disabling Auditing, a User needs adequate permissions. Please refer to the section 34.5 ACLS for Audit Management with regard to 'permissions'.

34.3 Enabling Auditing

Since policies define if events are recorded, in order to enable auditing for certain kinds of events the User needs to save certain policies with certain configurations. To save policies, follow the steps below:

1. Start the server. Login to the MQ dashboard.
2. Go to Audit Manager.
3. Go to Configure Audit Policies. Select Audit Events to be recorded such as Authentication, Authorization or Security Database Modification. By default all policies are passive.

4. Select **Action -> Modify Audit Storage Policy**. Details of the events recorded may be defined through this step, as shown below.



Policy Name	Event Type	Status	
Authorization Policy	Authorization Event	PASSIVE	☐
Security Database Modification Policy	Security Event	PASSIVE	☑
Authentication Policy	Authentication Event	PASSIVE	☐

Modify Audit Policy for : Security Database Modification Policy

Audit Event IDs	User Addition, User Deletion, Password Change, Group Addition, Group Deletion, Group Member Addition, Group Member Deletion, Grant Permission, Deny Permission, Revoke Permission, ACL Removal, ACLEntry Removal
Event Status	[✓ Success] Failure
Event Category	[✓ Info] Warn Error

The figure above illustrates recording the addition/deletion of events. This step can be repeated for each different event to be recorded.

Note: This step maybe repeated when a user wishes to edit a policy that is active,

34.4 Disabling Auditing

To disable auditing certain events, please refer to steps One through Four in 34.3 Enabling Auditing. Instead of modifying the policy, select:

De-activate Storage Policy.

This action deletes the storage policy.

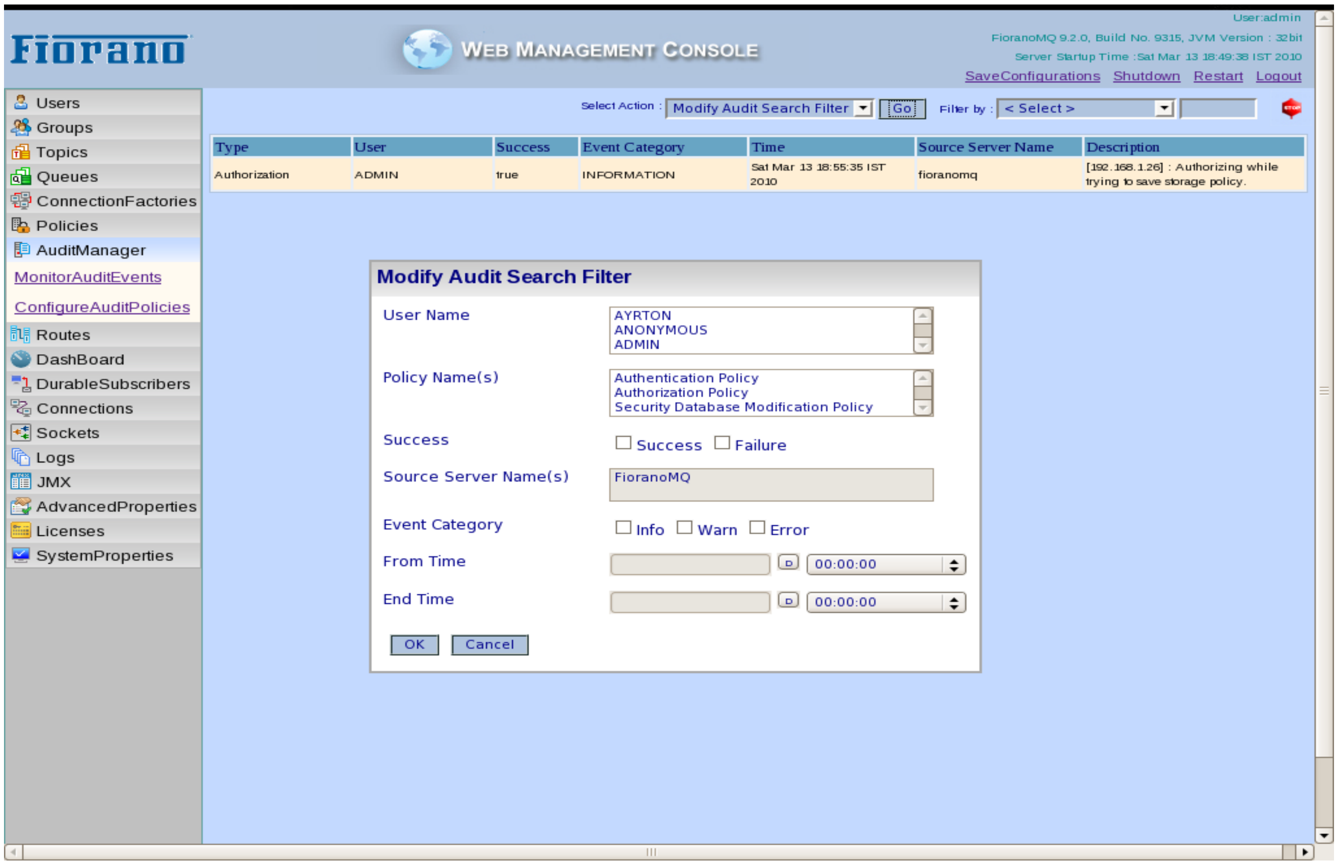
34.5 ACLS for Audit Management

By default, all User that form the Administrators group have permissions to save/edit/delete storage policies. In FioranoMQ there are no steps by which a User can assign permissions to non-administrator Users. If a user is to be allowed to save/edit/delete storage policies, that User must be added to 'administrators'.

34.6 Viewing Audit Events

To view the audit events refer toSteps One through Three in [34.3 Enabling Auditing](). And, select Monitor Audit Events -> Modify Audit Search Filter.

Though a User might wish that all events are recorded, not all events may need to be viewed. The level of importance of the events to be viewed can be set, as shown below.



Type	User	Success	Event Category	Time	Source Server Name	Description
Authorization	ADMIN	true	INFORMATION	Sat Mar 13 18:55:35 IST 2010	fioranomq	[192.168.1.26] : Authorizing while trying to save storage policy.

Modify Audit Search Filter

User Name	AYRTON ANONYMOUS ADMIN
Policy Name(s)	Authentication Policy Authorization Policy Security Database Modification Policy
Success	☐ Success ☐ Failure
Source Server Name(s)	FioranoMQ
Event Category	☐ Info ☐ Warn ☐ Error
From Time	00:00:00
End Time	00:00:00

34.7 Configuring File Store

By default, all Audit Management configurations persist/exist? in a file based storage.

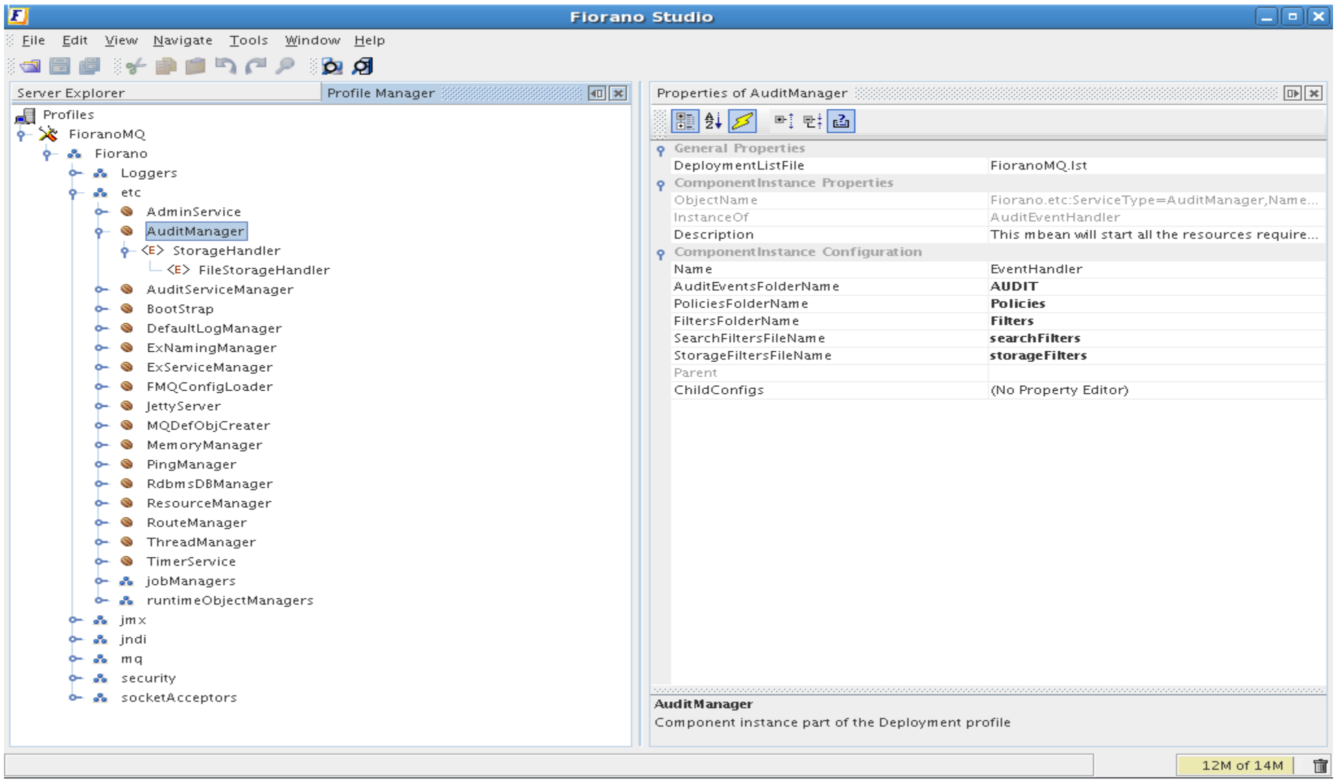
Certain parameters can be used to specify the relative directories in which AUDIT related information will is to be stored. These parameters are listed below.

- AuditEventsFolderName – This parameter is used to specify the working directory for audit management. The default directory path is $FMQ_DB_PATH/AUDIT.
- FiltersFolderName – This parameter is used to specify the name of the directory for storing audit event filters. The default directory path is $FMQ_DB_PATH/$AuditEventFolderName/FILTERS.
- SearchFiltersFileName – This parameter is used to specify the name of the file used to store search filters.
- StorageFiltersFileName - This parameter is used to specify the name of the file used to store storage filters.
- PoliciesFolderName – This parameter is used to specify the name of the directory for storing policies. The default directory path is $FMQ_DB_PATH/$AuditEventFolderName/POLICIES.
- FileName – This parameter is used to specify the name of the file in which the audit events are persisted.

These parameters may be configured offline in multiple ways such as though Studio, Config.cfg, and Config.xml file.

34.7.1. Through Studio

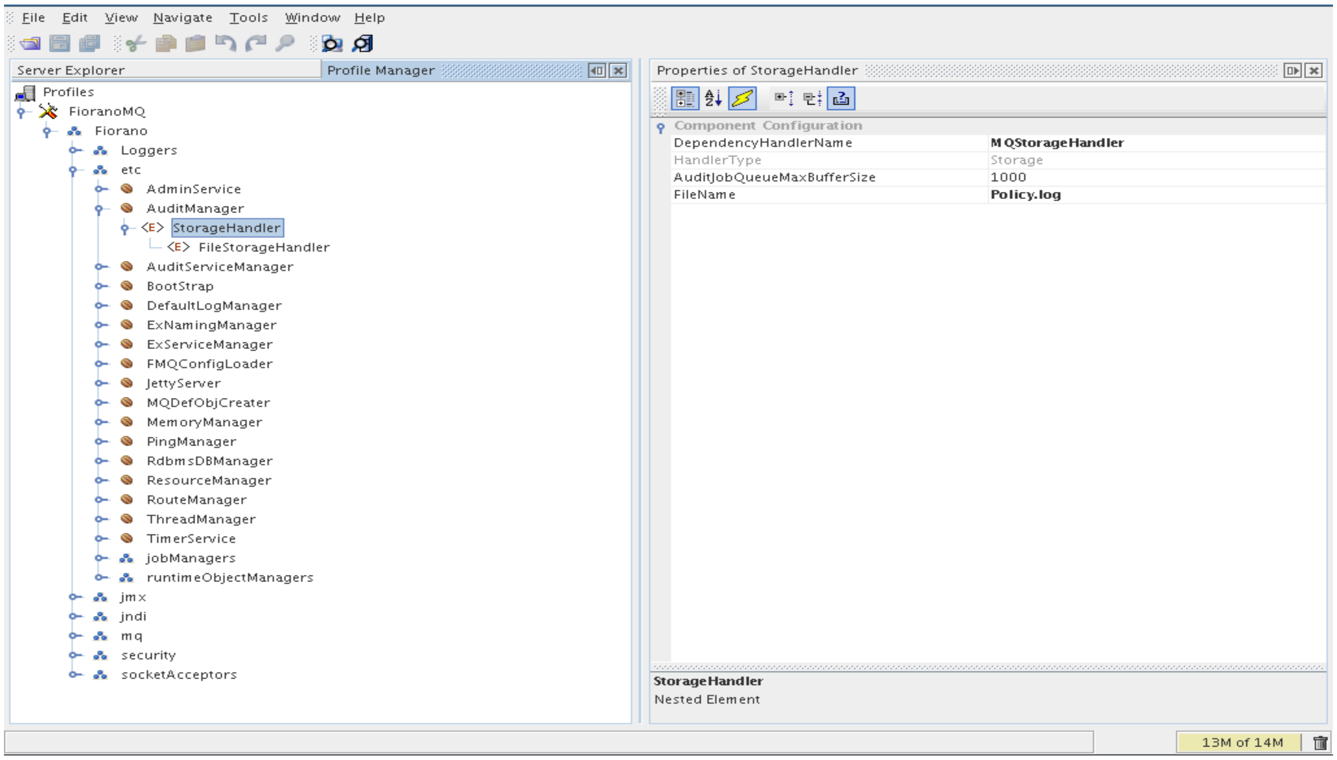
1. Open the required profile. Navigate to Fiorano -> etc -> Audit Manager



General Properties	
DeploymentListFile	FioranoMQ.lst
ComponentInstance Properties	
ObjectName	Fiorano.etc:ServiceType=AuditManager,Name...
InstanceOf	AuditEventHandler
Description	This mbean will start all the resources require...
ComponentInstance Configuration	
Name	EventHandler
AuditEventsFolderName	**AUDIT**
PoliciesFolderName	**Policies**
FiltersFolderName	**Filters**
SearchFiltersFileName	**searchFilters**
StorageFiltersFileName	**storageFilters**
Parent	
ChildConfigs	(No Property Editor)

AuditManager
Component instance part of the Deployment profile

2. Navigate to Fiorano -> etc -> Audit Manager -> StorageHandler



Component Configuration

DependencyHandlerName	MQStorageHandler
HandlerType	Storage
AuditJobQueueMaxBufferSize	1000
FileName	Policy.log

StorageHandler
Nested Element

34.7.2 Through Configs.cfg

Navigate to element:

ObjectName=Fiorano.etc:Name=AuditManager,ServiceType=AuditManager,type=config

ClassName=fiorano.audit.events.impl.config.AuditEventHandlerConfig

...

ObjectName=Fiorano.etc:Name=AuditManager,ServiceType=AuditManager,type=config

34.7.3 Through Configs.xml

Navigate to element:

<AuditEventHandler
ObjectName="Fiorano.etc:ServiceType=AuditManager,Name=AuditManager,type=config" ...>

...

</AuditEventHandler>

Chapter 35: Monitoring FioranoMQ Server

This chapter talks about monitoring the FioranoMQ message throughput statistics and getting notifications about the any special events configured.

35.1 Message Throughput

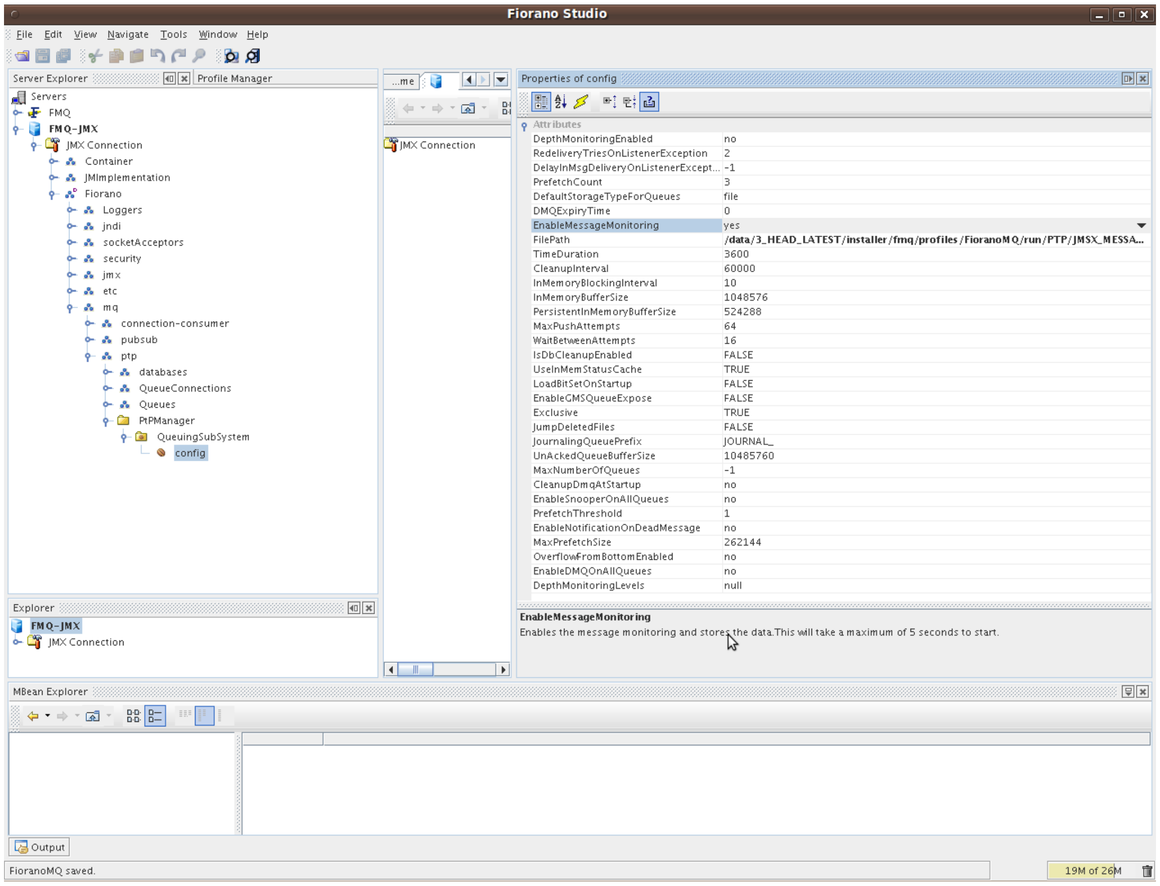
The FioranoMQ Server will not, by default, measure message throughput. However, a user can configure certain parameters to see related throughput statistics.

35.1.1 Server Side Configuration

Viewing Server Monitoring results:

Online

1. Login to FMQ-JMX in the **Server Explorer** pane by allocating the appropriate RMIConnector property values in the **Properties of FMQ-JMX** pane.
2. In FMQ-JMX, navigate to JMX-Connection**-->Fiorano-->mq-->ptp-->PtpManager->QueuingSubSystem->config** and go to the **Properties of QueuingSubSystem** pane as shown in the figure below.



Properties of config

Attributes

DepthMonitoringEnabled	no
RedeliveryTriesOnListenerException	2
DelayInMsgDeliveryOnListenerExcept...	-1
PrefetchCount	3
DefaultStorageTypeForQueues	file
DMQExpiryTime	0
EnableMessageMonitoring	yes
FilePath	/data/3_HEAD_LATEST/installer/fmq/profiles/FioranoMQ/run/PTP/JMSX_MESSA...
TimeDuration	3600
CleanupInterval	60000
InMemoryBlockingInterval	10
InMemoryBufferSize	1048576
PersistentInMemoryBufferSize	524288
MaxPushAttempts	64
WaitBetweenAttempts	16
IsDbCleanupEnabled	FALSE
UseInMemStatusCache	TRUE
LoadBitSetOnStartup	FALSE
EnableGMSQueueExpose	FALSE
Exclusive	TRUE
JumpDeletedFiles	FALSE
JournalingQueuePrefix	JOURNAL_
UnAckedQueueBufferSize	10485760
MaxNumberOfQueues	-1
CleanupDmqAtStartup	no
EnableSnooperOnAllQueues	no
PrefetchThreshold	1
EnableNotificationOnDeadMessage	no
MaxPrefetchSize	262144
OverflowFromBottomEnabled	no
EnableDMQOnAllQueues	no
DepthMonitoringLevels	null

EnableMessageMonitoring

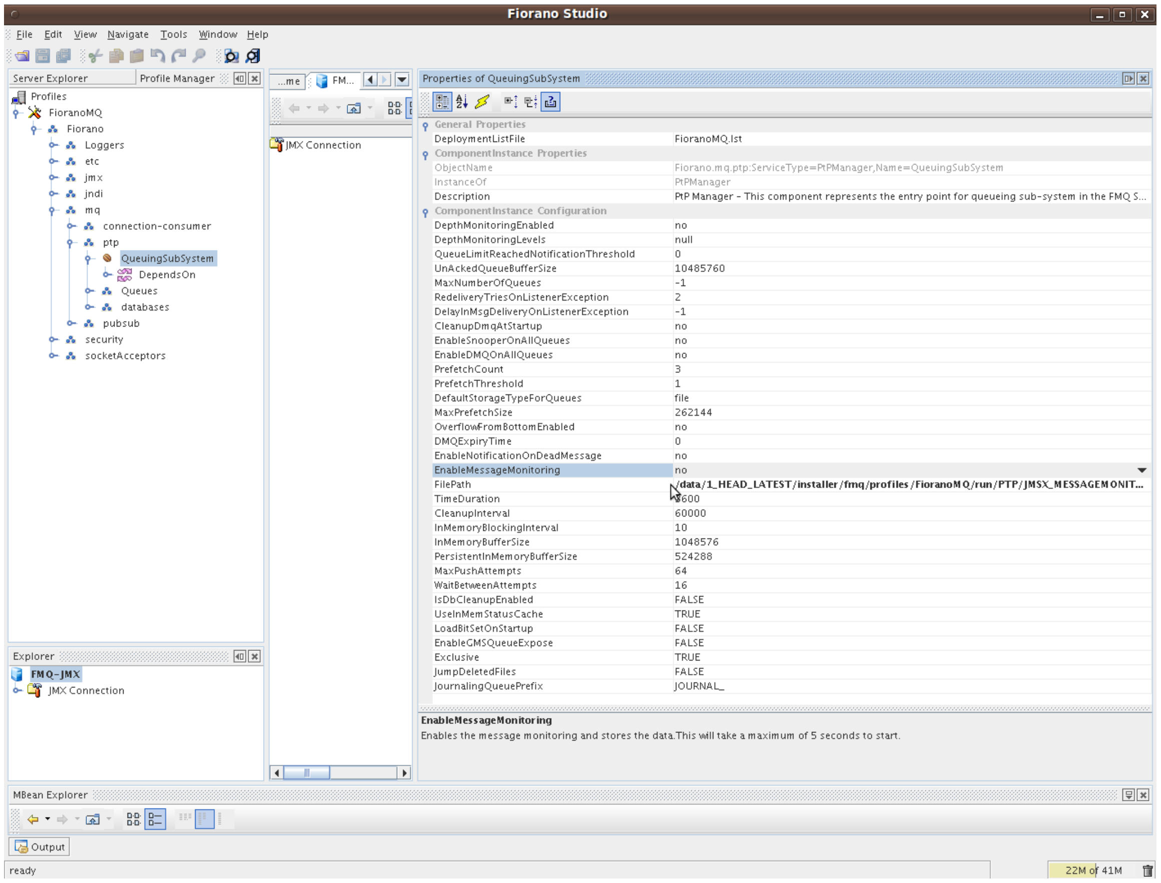
Enables the message monitoring and stores the data.This will take a maximum of 5 seconds to start.

FioranoMQ saved.

Offline

1. Right-click the **Profiles** node in the **Profile Manager** Explorer, and select the **Open** Profile option from the pop-up menu. In the **Select Profile Directory dialog**, select the **FioranoMQ** profile directory and click the **Open** button.

2. In Profile Manager, navigate to **FioranoMQ-->Fiorano-->mq-->ptp-->QueuingSubSystem** and go to the **Properties of QueuingSubSystem** pane as shown in the figure below



Properties of QueuingSubSystem

General Properties	
DeploymentListFile	FioranoMQ.lst
ComponentInstance Properties	
ObjectName	Fiorano.mq.ptp:ServiceType=PtPManager,Name=QueuingSubSystem
InstanceOf	PtPManager
Description	PtP Manager - This component represents the entry point for queueing sub-system in the FMQ S...
ComponentInstance Configuration	
DepthMonitoringEnabled	no
DepthMonitoringLevels	null
QueueLimitReachedNotificationThreshold	0
UnAckedQueueBufferSize	10485760
MaxNumberOfQueues	-1
RedeliveryTriesOnListenerException	2
DelayInMsgDeliveryOnListenerException	-1
CleanupDmqAtStartup	no
EnableSnooperOnAllQueues	no
EnableDMQOnAllQueues	no
PrefetchCount	3
PrefetchThreshold	1
DefaultStorageTypeForQueues	file
MaxPrefetchSize	262144
OverflowFromBottomEnabled	no
DMQExpiryTime	0
EnableNotificationOnDeadMessage	no
EnableMessageMonitoring	no
FilePath	/data/1_HEAD_LATEST/installer/fmq/profiles/FioranoMQ/run/PTP/JMSX_MESSAGEMONIT...
TimeDuration	3600
CleanupInterval	60000
InMemoryBlockingInterval	10
InMemoryBufferSize	1048576
PersistentInMemoryBufferSize	524288
MaxPushAttempts	64
WaitBetweenAttempts	16
IsDbCleanupEnabled	FALSE
UseInMemStatusCache	TRUE
LoadBitSetOnStartup	FALSE
EnableGMSQueueExpose	FALSE
Exclusive	TRUE
JumpDeletedFiles	FALSE
JournalingQueuePrefix	JOURNAL_

EnableMessageMonitoring
Enables the message monitoring and stores the data.This will take a maximum of 5 seconds to start.

Editing Properties

- **EnableMessageMonitoring**: Set this property to **yes** to enable the monitoring APIs. By default this property is set to **no**.

  For every monitored value there the initial value which is obtained when the message monitoring is enabled. This initial value is displayed at the end of the value in parenthesis.

- **FilePath**: Set this to the absolute location where the data is present. FIORANO_HOME environment variable needs to be set for this path for enabling proper location in the system.

  The Filepath can be changed at any instance. If the User has assigned a non-existing path, the Server tries to create the directory structure. However, changing the Filepath leads to loss of all data apart from initial values obtained when message monitoring is enabled. The Complete path is given in the Filepath so that the User can assign any location other than the run directory of FMQ server profile. If the location directory is not pointing to the FioranoMQ profile run directory, the clearing database does not remove the data. When the server is restarted the User can see the old data present in that directory. However, the initial values which are used for comparison are lost.

  Warning: The User should not open the files or directory present in the Filepath. This can lead to results that are misleading.

- **TimeDuration**: Sets the duration (last completed Time Interval) for which results are returned.

The Time Duration must be more than two seconds for better results.

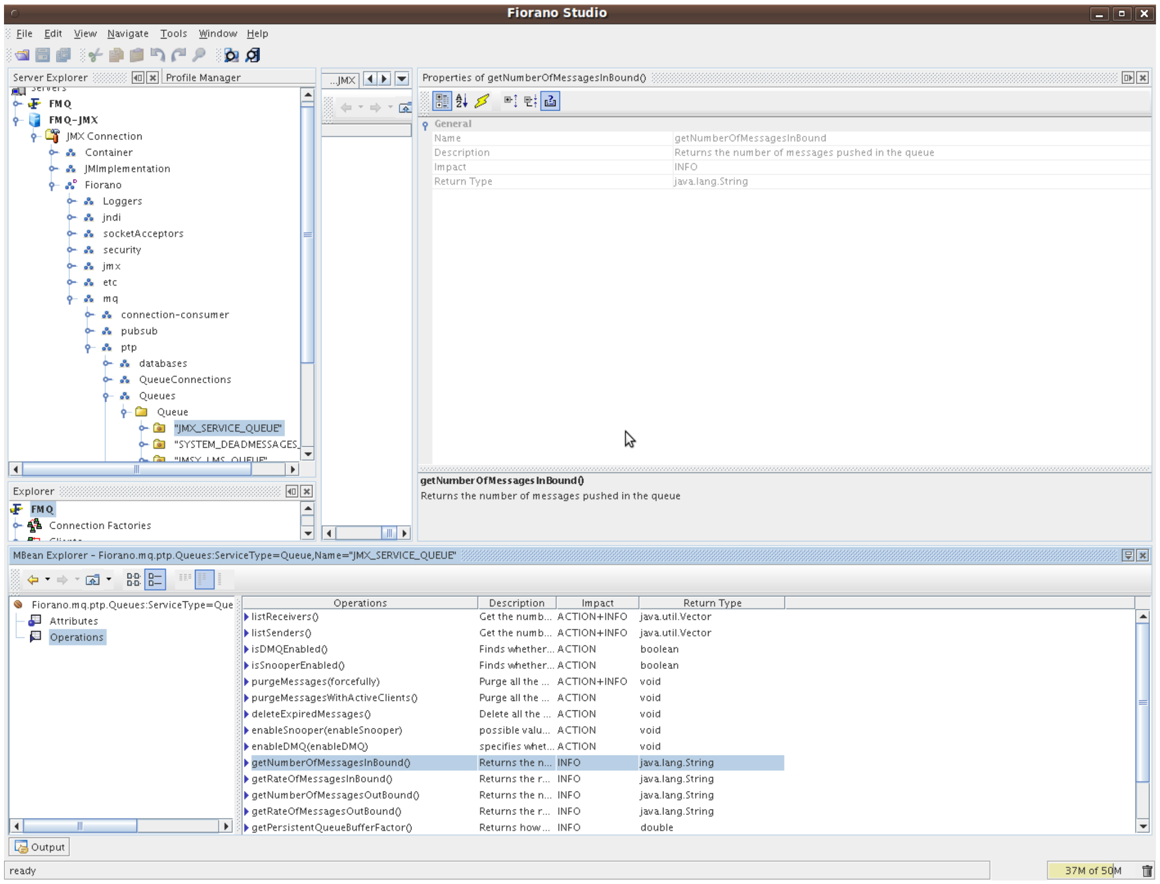
Note: The above three properties can be modified Online or Offline. The modification of these properties takes affect without restarting the Server.

Using Server Monitoring APIs of PTP Subsystem

Note: From FioranoMQ 9.5.1 onwards, in order to use JMX invocation, set the filePath parameter in the QueueSubsystem level to null. By doing so, FioranoMQ server will not dump any information in the message monitoring log files. This is done to avoid inconsistencies which led to decrease in values returned by getNumberOfMessagesInBound() when invoked through JMX.

The APIs listed below are to be added for Server monitoring. To view the APIs and the results:

1. Login to **FMQ-JMX** in the **Server Explorer** pane and navigate to **FMQ-JMX-> JMX Connection-> Fiorano-> mq-> ptp->Queues-> Queue-> PRIMARYQUEUE** (JMX_SERVICE_QUEUE). Now you can see the following APIs in the **MBean Exlporer** pane at the bottom of the window.
2. To see the result of an API, select **Operations** in the **Mbean** explorer pane. Select an API and right-click, and then click the **Invoke** option from the pop-up menu as shown in the figure below.
   - **getNumberOfMessagesInBound**
     This method returns the number of messages added to the queue of the previous TimeDuration in seconds.
   - **getRateOfMessagesInBound**
     This method returns the average number of messages added to the queue within one second in of the previous TimeDuration seconds.
   - **getNumberOfMessagesOutBound**
     This method returns the number of messages removed from the queue in the previous TimeDuration in seconds.
   - **getRateOfMessagesOutBound**
     This method returns the average number of messages removed from the queue within one second of the previous TimeDuration in seconds.



Properties of getNumberOfMessagesInBound()

General

Name	getNumberOfMessagesInBound
Description	Returns the number of messages pushed in the queue
Impact	INFO
Return Type	java.lang.String

getNumberOfMessagesInBound()
Returns the number of messages pushed in the queue

MBean Explorer - Fiorano.mq.ptp.Queues:ServiceType=Queue,Name="JMX_SERVICE_QUEUE"

Operations	Description	Impact	Return Type
listReceivers()	Get the numb...	ACTION+INFO	java.util.Vector
listSenders()	Get the numb...	ACTION+INFO	java.util.Vector
isDMQEnabled()	Finds whether...	ACTION	boolean
isSnooperEnabled()	Finds whether...	ACTION	boolean
purgeMessages(forcefully)	Purge all the ...	ACTION+INFO	void
purgeMessagesWithActiveClients()	Purge all the ...	ACTION	void
deleteExpiredMessages()	Delete all the ...	ACTION	void
enableSnooper(enableSnooper)	possible valu...	ACTION	void
enableDMQ(enableDMQ)	specifies whet...	ACTION	void
getNumberOfMessagesInBound()	Returns the n...	INFO	java.lang.String
getRateOfMessagesInBound()	Returns the r...	INFO	java.lang.String
getNumberOfMessagesOutBound()	Returns the n...	INFO	java.lang.String
getRateOfMessagesOutBound()	Returns the r...	INFO	java.lang.String
getPersistentQueueBufferFactor()	Returns how ...	INFO	double

- **countPendingMessages**
  This method returns the number of messages present in the queue at any given second.

Using Server Monitoring APIs of PubSub Subsystem

Note: From FioranoMQ 9.5.1 onwards, in order to use JMX invocation, set the filePath parameter in the TopicSubsystem level to null. By doing so, FioranoMQ server will not dump any information in the message monitoring log files. This is done to avoid inconsistencies which led to decrease in values returned by getNumberOfMessagesInBound() when invoked through JMX.

The following APIs are added at the Topic level.

1. Login to **FMQ-JMX** in the **Server Explorer** pane and navigate to **FMQ-JMX-> JMX Connection-> Fiorano-> mq-> pubsub-> topics->topic-> PRIMARYTOPIC**. It is possible to view APIs in the **MBean Exlporer** pane at the bottom of the window.
2. To see an API select **Operations** in the Mbean explorer and right-click one of the APIs .Click the **Invoke** option from the pop-up menu.
   - **getNumberOfMessagesInBound**
     This method returns the number of messages added to the queue in the last TimeDuration seconds.

- **getRateOfMessagesInBound**
  This method returns the average number of messages added to the queue within one second of the precious TimeDuration seconds.

- **getMessageMonitoringData**
  This method returns data in tabular form containing all the monitored data for a particular queue such as: **NumberOfMessagesInBound, RateOfMessagesInBound**

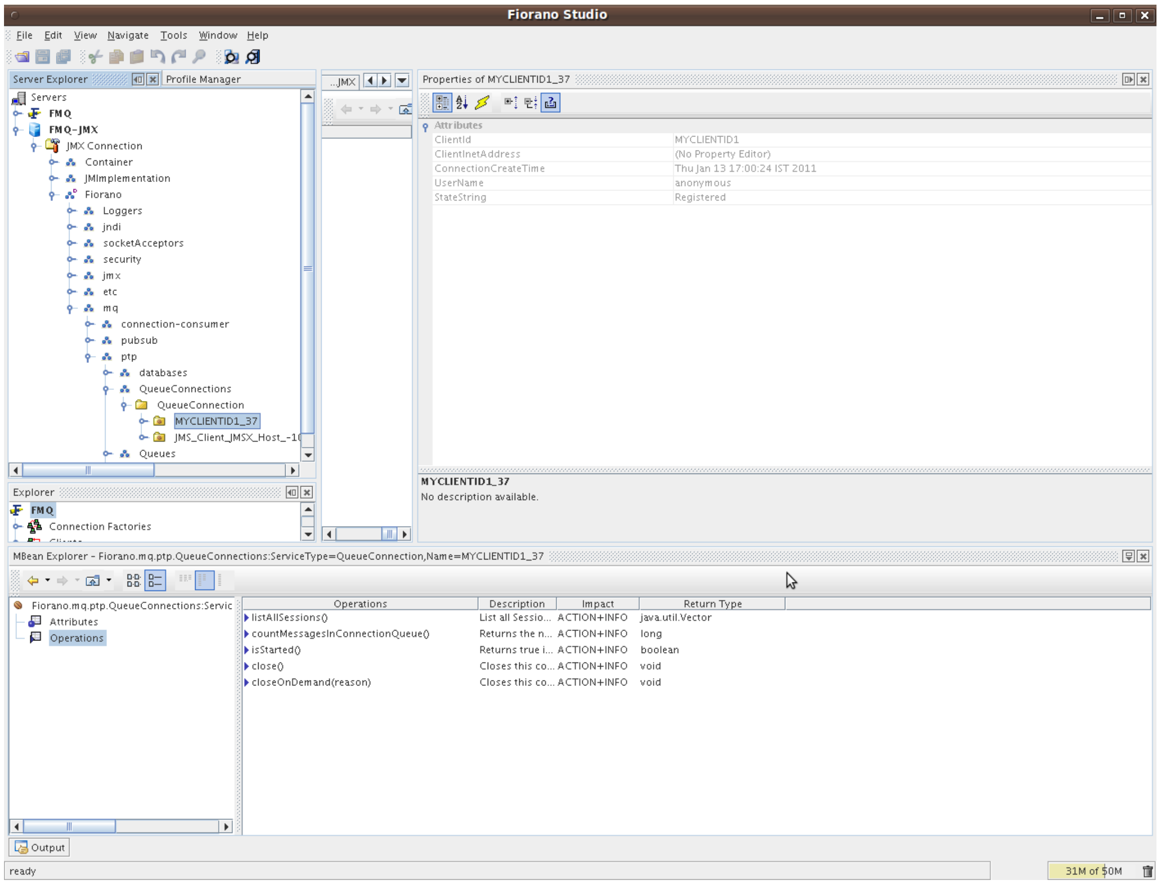
Using Server Monitoring APIs of QueueConnection

The APIs listed below are added to the QueueConnection level.

1. Login to **FMQ-JMX** in the **Server Explorer** pane and navigate to **FMQ-JMX-> JMX Connection-> Fiorano-> mq-> ptp-> QueueConnection -> PRIMARYQUEUE**. It is possible to view the APIs in the **MBean Exlporer** pane at the bottom of the window.

2. To view an API, right-click and choose the API. Click the **Invoke** option from the pop-up menu as shown in the figure below.

   - **countMessagesInConnectionQueue**
     This API returns the number of messages present in the connection queue buffer.



Properties of MYCLIENTID1_37

Attributes	
ClientId	MYCLIENTID1
ClientInetAddress	(No Property Editor)
ConnectionCreateTime	Thu Jan 13 17:00:24 IST 2011
UserName	anonymous
StateString	Registered

MYCLIENTID1_37
No description available.

MBean Explorer - Fiorano.mq.ptp.QueueConnections:ServiceType=QueueConnection,Name=MYCLIENTID1_37

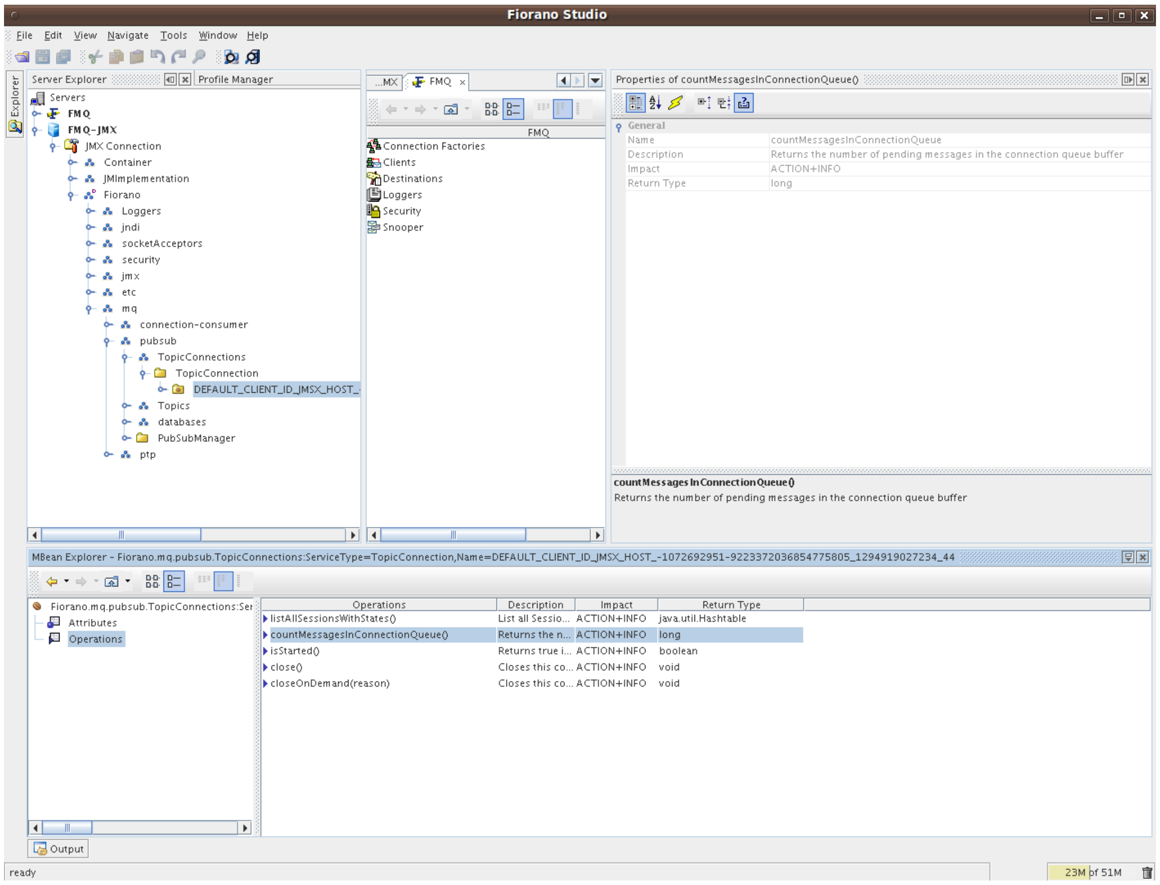
Fiorano.mq.ptp.QueueConnections:Servic
Attributes
Operations

Operations	Description	Impact	Return Type
listAllSessions()	List all Sessio...	ACTION+INFO	java.util.Vector
countMessagesInConnectionQueue()	Returns the n...	ACTION+INFO	long
isStarted()	Returns true i...	ACTION+INFO	boolean
close()	Closes this co...	ACTION+INFO	void
closeOnDemand(reason)	Closes this co...	ACTION+INFO	void

Using Server Monitoring APIs of TopicConnection

The following API was added at topic connection level.

- **countMessagesInConnectionQueue**
  This API returns the number of messages present in the connection buffer.


Fiorano Studio
File Edit View Navigate Tools Window Help
Explorer
Server Explorer
Profile Manager
...MX
FMQ
Properties of countMessagesInConnectionQueue()
Output

Servers
- FMQ
- FMQ-JMX
  - JMX Connection
    - Container
    - JMImplementation
    - Fiorano
      - Loggers
      - jndi
      - socketAcceptors
      - security
      - jmx
      - etc
      - mq
        - connection-consumer
        - pubsub
          - TopicConnections
            - TopicConnection
              - DEFAULT_CLIENT_ID_JMSX_HOST_
          - Topics
          - databases
          - PubSubManager
        - ptp

FMQ
- Connection Factories
- Clients
- Destinations
- Loggers
- Security
- Snooper

General

Name	countMessagesInConnectionQueue
Description	Returns the number of pending messages in the connection queue buffer
Impact	ACTION+INFO
Return Type	long

countMessagesInConnectionQueue()
Returns the number of pending messages in the connection queue buffer

MBean Explorer - Fiorano.mq.pubsub.TopicConnections:ServiceType=TopicConnection,Name=DEFAULT_CLIENT_ID_JMSX_HOST_-1072692951-9223372036854775805_1294919027234_44

Fiorano.mq.pubsub.TopicConnections:Ser
- Attributes
- Operations

Operations	Description	Impact	Return Type
listAllSessionsWithStates()	List all Sessio...	ACTION+INFO	java.util.Hashtable
countMessagesInConnectionQueue()	Returns the n...	ACTION+INFO	long
isStarted()	Returns true i...	ACTION+INFO	boolean
close()	Closes this co...	ACTION+INFO	void
closeOnDemand(reason)	Closes this co...	ACTION+INFO	void

ready 23M of 51M

35.1.2. Client Side Configuration

To monitor destinations 'EnableMessageMonitoring' the flag must be set to 'Yes'.

When a Producer or Consumer is created on FioranoMQ Server, hierarchical runtime MBeans are created on the Server. These MBeans required that methods are enabled.

1. Client side Requirement:

   Client ConnectionID, SessionID, ProducerID/ConsumerID should be set to identify instances on the Server. ConnectionID can be set using JMS apis SessionID, ProducerID/ConsumerID are set using the parameters listed below before connection creation is added. (This Java or equivalent code should be added to the client Java or correspoding code.)

System.setProperty("FIORANO_SESSIONID", "PublisherSession");

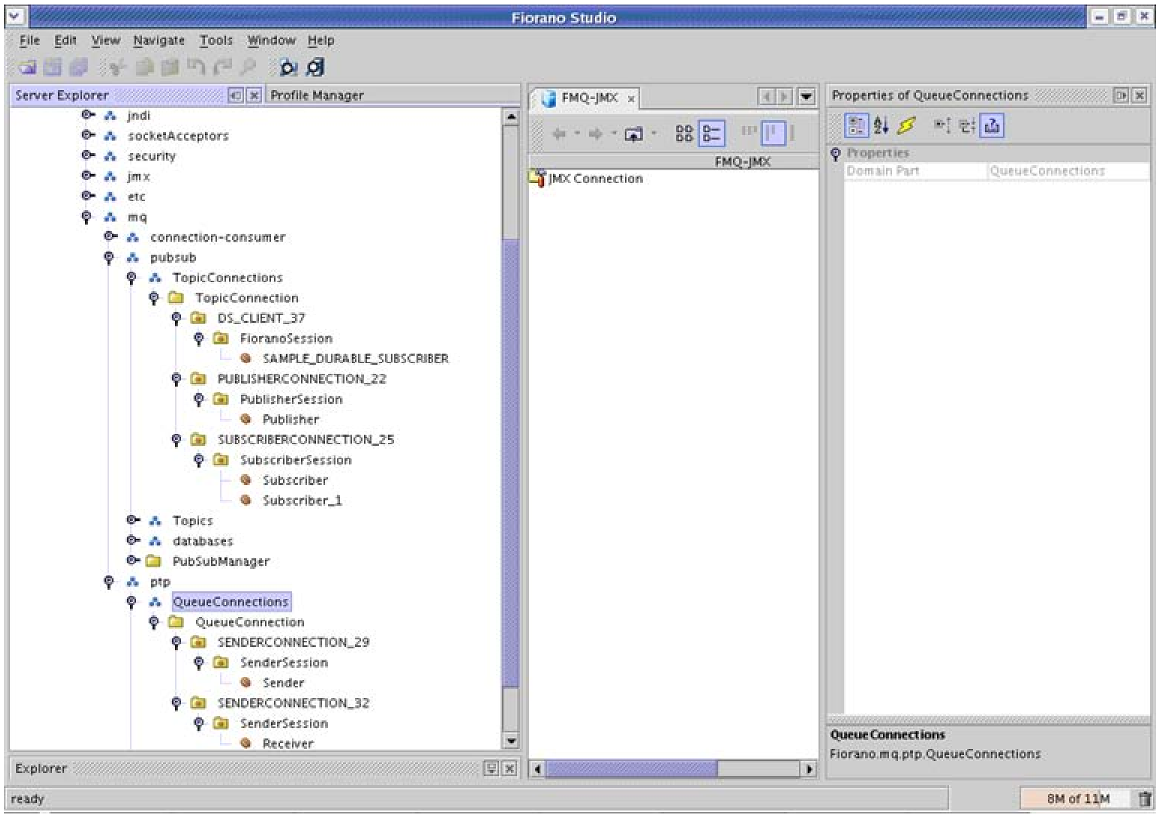
System.setProperty("FIORANO_PRODUCERID","Publisher");

System.setProperty("FIORANO_CONSUMERID", "Subscriber");

All properties need not be added. Publisher creation "FIORANO_CONSUMERID" need not be set.

2. Consequences in Server side:

The properties are required to identify a client on a Server. If these properties are not set then the default values are used. For a durable subscriber ConsumerID need not be set and the one passed during creation will be used as its name. If more than one instance of a session involving producer/consumer are used then the serial number will be appended to the property. If a user creates more than one connection in the same JVM then the properties set before the creation of the connection will be the ones to take effect.



```
jndi
socketAcceptors
security
jmx
etc
mq
    connection-consumer
    pubsub
        TopicConnections
            TopicConnection
                DS_CLIENT_37
                    FioranoSession
                        SAMPLE_DURABLE_SUBSCRIBER
                PUBLISHERCONNECTION_22
                    PublisherSession
                        Publisher
                SUBSCRIBERCONNECTION_25
                    SubscriberSession
                        Subscriber
                        Subscriber_1
        Topics
        databases
        PubSubManager
    ptp
        QueueConnections
            QueueConnection
                SENDERCONNECTION_29
                    SenderSession
                        Sender
                SENDERCONNECTION_32
                    SenderSession
                        Receiver
```

FMQ-JMX
JMX Connection

Properties

Domain Part	QueueConnections

Queue Connections
Fiorano.mq.ptp.QueueConnections

The image above displays how a User creates a QueueSender, QueueReceiver, Publisher, DurableSubscriber and two subscribers in the same session. For every ConnectionID the hashCode of that connection will be appended to it.

In the figure under SUBSCRIBERCONNECTION_25 there two subscribers are created. The first subscriber is named by the User and the second '1' appended to it.

APIs added at Destination level in pubsub:

- getNumSubscribedMsgs:

  This API will return the number of messages that are received by the subscriber in the configInterval time configured in TopicSubSystem properties. DurableSubscriber message count starts when the durable subscriber is initiated. If the durable subscriber is closed and started again then the message count will be zero.

- getNumPublishedMsgs:

This APII returns the number of messages that are published by the publisher in the configInterval time configured in TopicSubSystem properties.

APIs added at Destination level in PTP:

- getNumReceivedMsgs:

  This returns the number of messages that are received by the receiver in the configInterval time configured in QueuingSubSystem properties.

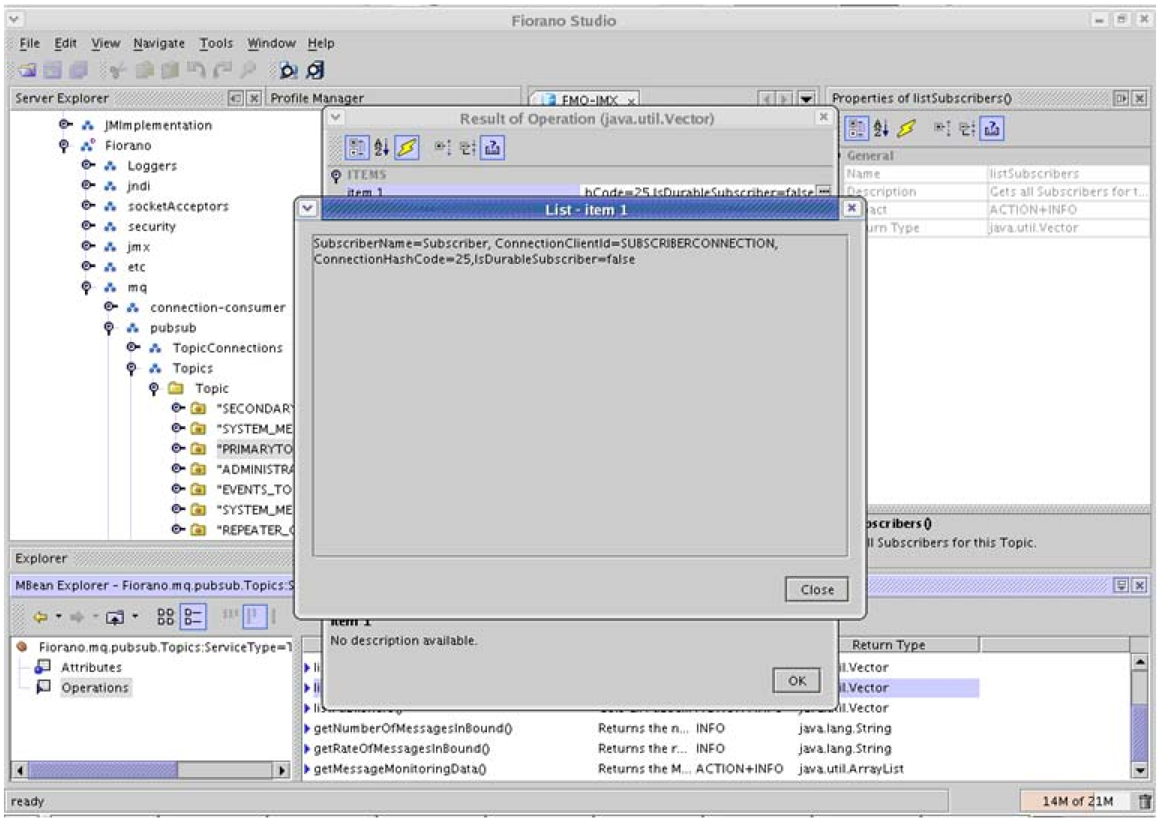
- getNumSendMsgs:

  This returns the number of messages that are sent by the sender in the configInterval time configured in QueuingSubSystem properties.

  Few more JMX APIs are added at the destination level to list the number of Producers and Consumers within a destination.

- listSubscribers:

  This returns the list of subscribers created on a particular topic.



```
SubscriberName=Subscriber, ConnectionClientId=SUBSCRIBERCONNECTION,
ConnectionHashCode=25,IsDurableSubscriber=false
```

getNumberOfMessagesInBound()	Returns the n...	INFO	java.lang.String
getRateOfMessagesInBound()	Returns the r...	INFO	java.lang.String
getMessageMonitoringData()	Returns the M...	ACTION+INFO	java.util.ArrayList

The figure shows 'SubscriberName' the MBeanName,'ConnectionClientID' which is the connection ID of the subscriber set from the client and 'ConnectionHashCode' is the hashcode. These parameters are required for identifying the connection. 'IsDurableSubscriber' indicates whether a subscriber is durable.

- listPublishers:

  This returns the list of publishers created on that particular topic.

- listReceivers:

This returns the list of receivers created on that particular queue.

- listSenders:

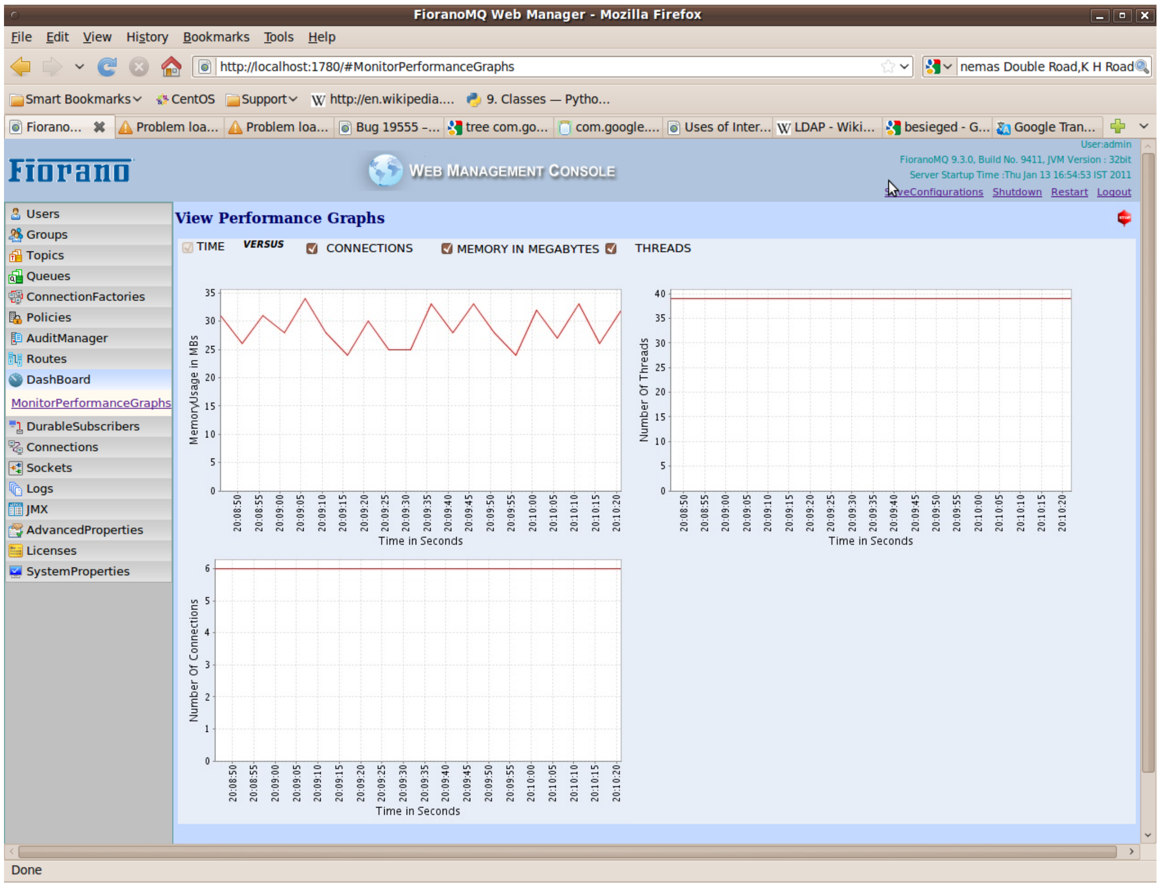
  This returns the list of senders created on that particular queue.

35.1.3 Performance graphs

Performance graph showing connections, memory usage and threads can be seen through Web Management console. Steps are:

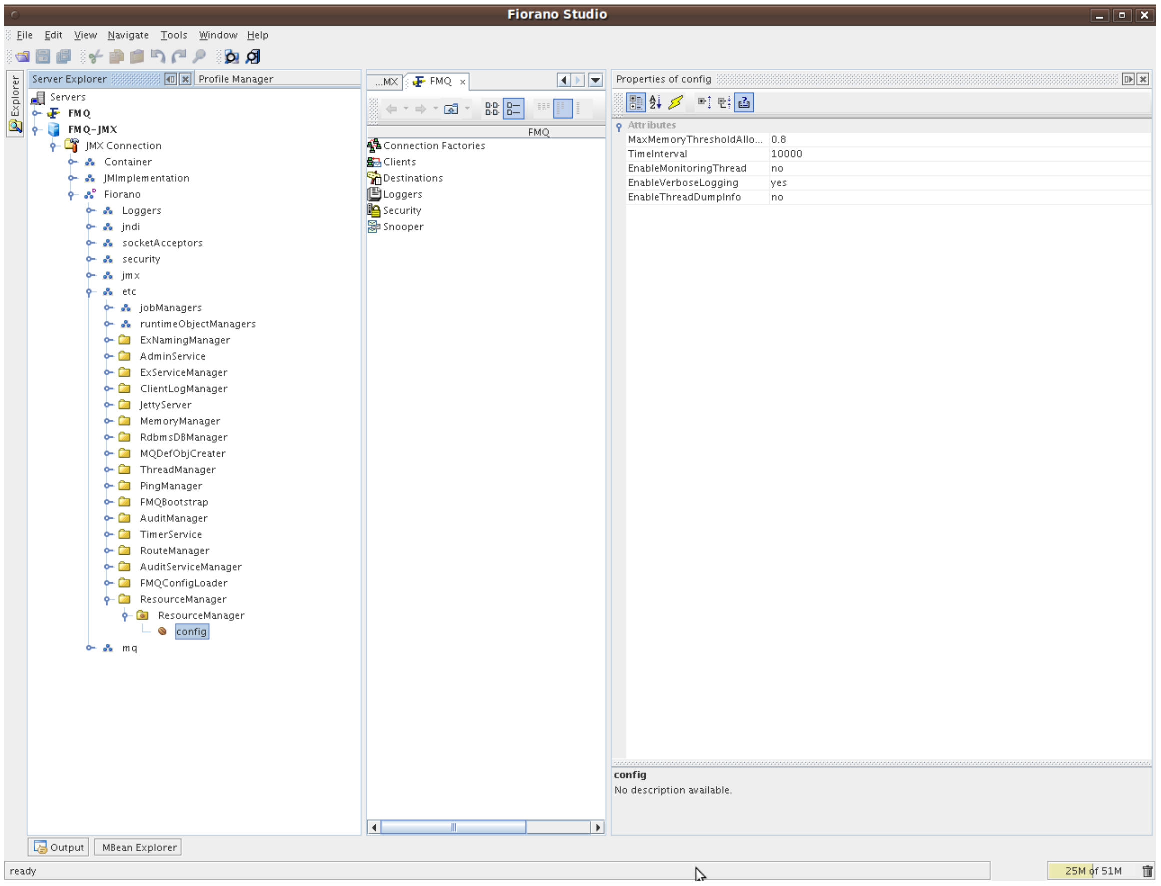
1. Log-in to Web Management console.
2. From Left hand side pane click on dashboard → select MonitorPerformanceGraphs. Options regarding performance graphs will appear on right hand side area.
3. Select among options:
   a. CONNECTIONS
   b. MEMORY IN MEGABYTES
   c. THREADS

Corresponding graph will be plotted against time. Following figure shows a screen shot of this view.



35.1.4 Logging server Monitoring information

Monitoring Thread can be enabled for FMQ server by setting “EnableMonitoringThread” parameter. Following screen shot shows how to enable it form FMQ-JMX log-in. Log-in to FMQ-JMX through Studio and follow tree FMQ-**JMX-> JMX Connection-> Fiorano->etc->Resource Manager->ResourceManager->config**.



Properties of config

Attributes

MaxMemoryThresholdAllo...	0.8
TimeInterval	10000
EnableMonitoringThread	no
EnableVerboseLogging	yes
EnableThreadDumpInfo	no

config
No description available.

Moreover through Web Management Console this parameter can be configured. For doing so through Web Management Console log-in to Web management console-> From left hand side pane click on AdvancedProperties → select ConfigureAdvancedProperties → From Right hand side tree options expand “ResourceManager Properties” → invoke “EnableMonitoringThread” operation and enable it.

After enabling this parameter following monitoring information will be logged in file “FioranoMQ_home\fmq\profiles\<profile name>\run\logs\monitor.txt”:

- Number of Connections
- Number of Sessions
- Number of Producers
- Number of Consumers
- Number of Activated Destinations
- Total Number of JVM Thread
- Used Memory of the Server

This information will be noted on a regular interval.

35.2 Depth Monitoring

A User can get depth or pending message count of a queue from the apis available at queue level. A User can view the pending message using the administration connection or the existing JMX Apis. If a User wants to capture events when the depth of a queue crosses a certain level, the User can go through this section to understand how to configure the Server to receive 'depths' notification. *Currently there is only the Server send depth notification for queues available.*

35.2.1 Configuring Through Configuration files

To configure 'depths' of a queue, the user needs to enable depth monitoring at the Server level.

To enable depth monitoring at the Server level.

1. Open Configs.cfg file under the conf folder of the Server profile.
2. Navigate to Queue Settings. Set DepthMonitoringEnabled to TRUE.

# Queue Settings

ObjectName=Fiorano.mq.ptp:ServiceType=PtPManager,Name=QueuingSubSystem,type=confi g

ClassName=fiorano.jms.ex.ptp.config.ExPTPManagerConfig

DepthMonitoringEnabled=true

ObjectName=Fiorano.mq.ptp:ServiceType=PtPManager,Name=QueuingSubSystem,type=confi g

Once this flag is set to **TRUE**, depth monitoring is enabled at the Server level. The user can now enable depth monitoring for that particular queue.

To enable depth monitoring for the primary queue:

1. Open **Configs.cfg** file and navigate to **primary queue settings**.
2. Change DepthMonitoringEnabled to **TRUE**.

ObjectName=Fiorano.mq.ptp.Queues:Name="PRIMARYQUEUE",ServiceType=Queue,type=con fig

ClassName=fiorano.jms.common.config.QueueConfig

DepthMonitoringEnabled=true

ObjectName=Fiorano.mq.ptp.Queues:Name="PRIMARYQUEUE",ServiceType=Queue,type=con fig Depth Monitoring is enabled for the primary queue.

A User can configure the depth levels so that notifications are raised. To configure Depth levels, go to **queue settings** and change **DepthMonitoringLevels**. The value of DepthMonitoringLevels should be integers separated by commas. A sample value will be 'DepthMonitoringLevels=10,20,30'. This depth level is applicable to all queues. All queues will take the Depth levels set at Queue settings if individual depth levels are not set. If a User wishes to set different levels for different queues, the User needs to change these settings for all queue levels.

To receive notifications, a User needs to register the JMX notification listener on the Server. The file **JMXNotifications.java** will register notification listeners to the Server is present at **fmq/samples/JMX**. A User needs to register the listener to Queue Event Manager:

'Fiorano.jmx.notifications:ServiceType=EventManager,Name=QueueEventManager'

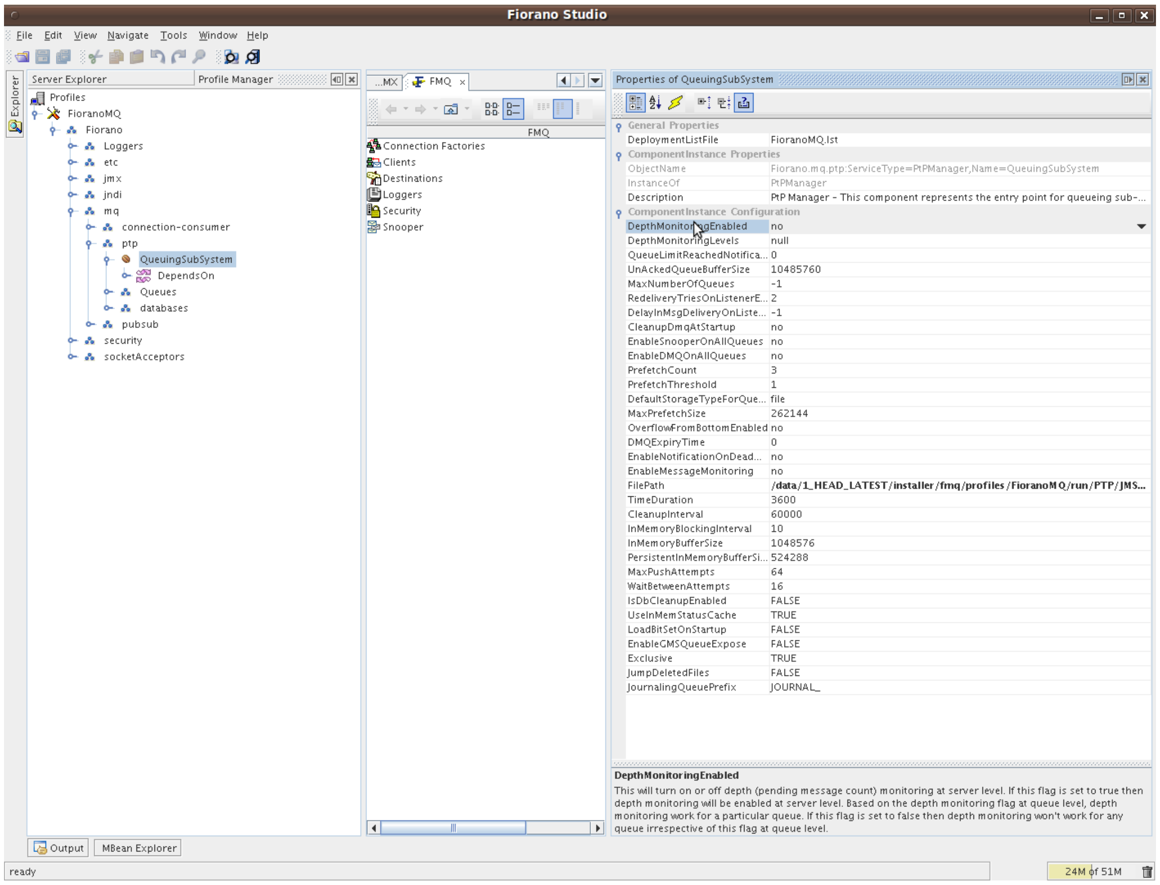
The above command receives notifications when the queue depth crosses the values defined by DepthMonitoringLevels.

35.2.2 Configuring Through Studio

1. Right-click the **Profiles** node in the **Profile Manager** Explorer, and select the **Open** Profile option from the pop-up menu.
2. In the **Select Profile Directory dialog**, select the **FioranoMQ** profile directory and click the **Open** button.

For enabling depth monitoring at server level:

In Profile Manager, navigate to FioranoMQ-->Fiorano-->mq-->ptp-->QueuingSubSystem and go to the Properties of QueuingSubSystem pane as shown in the figure below.



Properties of QueuingSubSystem

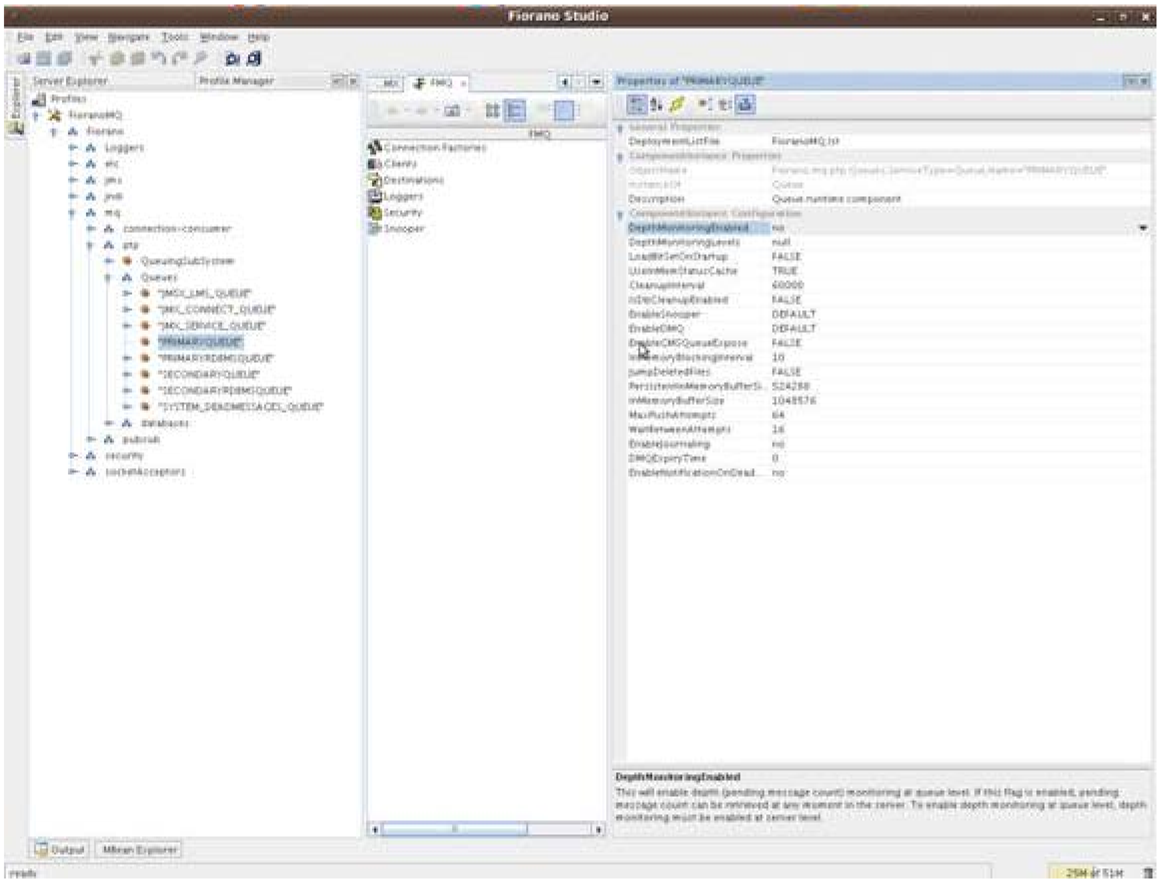
General Properties	
DeploymentListFile	FioranoMQ.lst
ComponentInstance Properties	
ObjectName	Fiorano.mq.ptp:ServiceType=PtPManager,Name=QueuingSubSystem
InstanceOf	PtPManager
Description	PtP Manager - This component represents the entry point for queueing sub-...
ComponentInstance Configuration	
DepthMonitoringEnabled	no
DepthMonitoringLevels	null
QueueLimitReachedNotifica...	0
UnAckedQueueBufferSize	10485760
MaxNumberOfQueues	-1
RedeliveryTriesOnListenerE...	2
DelayInMsgDeliveryOnListe...	-1
CleanupDmqAtStartup	no
EnableSnooperOnAllQueues	no
EnableDMQOnAllQueues	no
PrefetchCount	3
PrefetchThreshold	1
DefaultStorageTypeForQue...	file
MaxPrefetchSize	262144
OverflowFromBottomEnabled	no
DMQExpiryTime	0
EnableNotificationOnDead...	no
EnableMessageMonitoring	no
FilePath	**/data/1_HEAD_LATEST/installer/fmq/profiles/FioranoMQ/run/PTP/JMS...**
TimeDuration	3600
CleanupInterval	60000
InMemoryBlockingInterval	10
InMemoryBufferSize	1048576
PersistentInMemoryBufferSi...	524288
MaxPushAttempts	64
WaitBetweenAttempts	16
IsDbCleanupEnabled	FALSE
UseInMemStatusCache	TRUE
LoadBitSetOnStartup	FALSE
EnableGMSQueueExpose	FALSE
Exclusive	TRUE
JumpDeletedFiles	FALSE
JournalingQueuePrefix	JOURNAL_

DepthMonitoringEnabled

This will turn on or off depth (pending message count) monitoring at server level. If this flag is set to true then depth monitoring will be enabled at server level. Based on the depth monitoring flag at queue level, depth monitoring work for a particular queue. If this flag is set to false then depth monitoring won't work for any queue irrespective of this flag at queue level.

For enabling depth monitoring at particular queue:

In Profile Manager, navigate to FioranoMQ-->Fiorano-->mq-->ptp-->Queues-> >particular queue> and go to the Properties of selected queue pane as shown in the figure below



Properties of "PRIMARYQUEUE"

General Properties	
DeploymentListFile	FioranoMQ.lst
ComponentInstance Properties	
ObjectName	Fiorano.mq.ptp:ServiceType=Queue,Name="PRIMARYQUEUE"
InstanceOf	Queue
Description	Queue runtime component
ComponentInstance Configuration	
DepthMonitoringEnabled	no
DepthMonitoringLevels	null
LoadBitSetOnStartup	FALSE
UseInMemStatusCache	TRUE
CleanupInterval	60000
IsDbCleanupEnabled	FALSE
EnableSnooper	DEFAULT
EnableDMQ	DEFAULT
EnableDMQQueueExpose	FALSE
InMemoryBlockingInterval	10
JumpDeletedFiles	FALSE
PersistentInMemoryBufferSi...	524288
InMemoryBufferSize	1048576
MaxPushAttempts	64
WaitBetweenAttempts	16
EnableJournaling	no
DMQExpiryTime	0
EnableNotificationOnDead...	no

DepthMonitoringEnabled

This will enable depth (pending message count) monitoring at queue level. If this flag is enabled, pending message count can be retrieved at any moment in the server. To enable depth monitoring at queue level, depth monitoring must be enabled at server level.

Chapter 36: Route Configuration

A Route enables the transport of messages between destinations. A message arriving on one destination can be made available to another by adding a route between the two destinations.

36.1 Parameters to Configure

- **UpperLimitOnConnectionSharing**: Specifies the maximum number of routes that will share a single JMS connection.
- **MaxCreateSessionTries**: Specifies the maximum tries to make while creating JMS objects.
- **MaxTopicBuffer**: Route level upper limit for message buffer size in bytes.
- **UseConnectionSharing**: Specifies whether connection sharing by routes is switched ON.

36.1.2 Parameters Used While Adding Routes

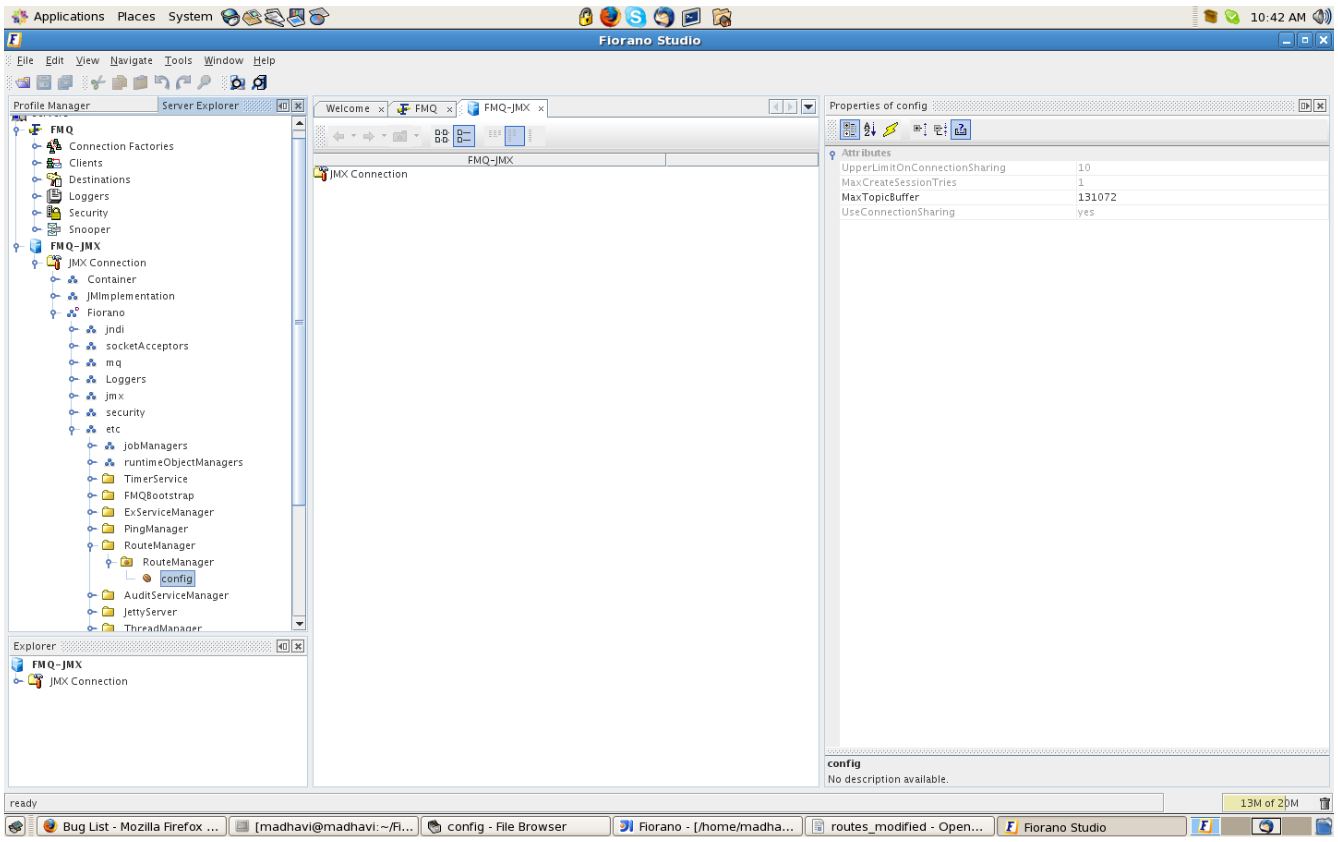
- **Guid**: Unique identifier for the route. No two routes can have the same id. If same id is specified for two routes, route creation will fail
- **SourceServer/Address**: Address of the server on which route will be created. Host name/IPAddress need to be specified here. For example, 'localhost'.
- **SourceServer/Port**: RMI port of the server on which this route will be created.
- **SourceServer/Destination**: Name of the destination from which this route will be created.
- **SourceServer/UserName**: Name of the user which will be used for authentication and route creation.
- **SourceServer/Password**: Password of the 'UserName' which will be used for authentication and route creation.
- **TargetServer/CFName**: TargetConnectionFactory in the target server which will be used to create JMSConnection.
- **TargetServer/Destination**: Target destination to which messages will be transferred on the target server.
- **TargetServer/UserName**: Name of the user which will be used for authentication and route creation.
- **TargetServer/Password**: Password of the 'UserName' which will be used for authentication and route creation.
- **TargetServer/ClientID**: Unique identification for the JMSConnection created on the target server. This id will be used to transfer messages, when the connection with the server is lost and connected.
- **IsDurable**: Boolean states whether durable subscriber should be created on the source server.

- **DurableSubscriber** id: Unique id for a durable subscriber on this connection. If IsDurable is set to true, then this property should be specified. This id will be used to identify a particular durable subscriber.
- **MessageSelector**: JMS Message selector in the route. If selector is not null, only selected messages will be picked up from the source destination. If message selector is specified as null, then all messages are picked from the source server.
- **EnableDMQOnRoute**: Enabled dead message queues on the route leads to storing expired messages in the DeadMessageQueue. If this value is set as 'no', expired messages are deleted from the server.
- **IsJMSReplyToAsTarget** : Boolean indicating whether target destination should be set dynamically based on the incoming message's JMSReplyTo header property. This is useful in the case of request-reply
- **MsgPropertyForTargetCF** : JMS message property used to set TargetCF, in case isJMSReplyToAsTarget is true and no Target CF is specified

36.2 Configuring Parameters Through Fiorano Studio

To configure parameters using Fiorano Studio, follow the steps below:

1. Run the fmq server and open Fiorano Studio.
2. Login to FMQ-JMX and navigate to Fiorano->etc->RouteManager->RouteManager ->config.
3. Clicking on the config displays the parameters that can be configured like MaxTopicBuffer, NumberOfRoutesExistingInServer, MaxCreateSessionTries, and so on. Following diagram shows the parameters to be configured using Studio



Properties of config

Attributes	
UpperLimitOnConnectionSharing	10
MaxCreateSessionTries	1
MaxTopicBuffer	131072
UseConnectionSharing	yes

config
No description available.

36.2.1 List/Add/Remove Routes

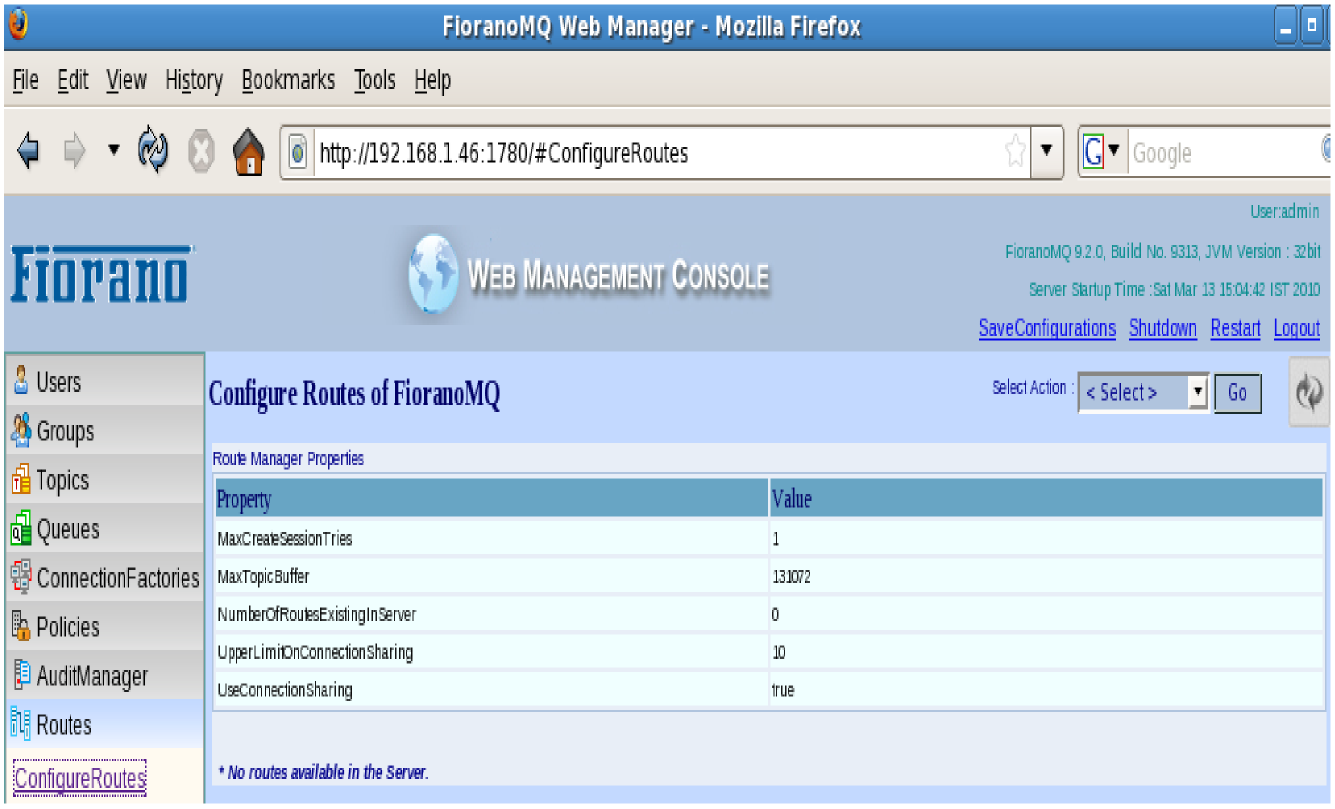
To List/Add/Remove routes, navigate to Fiorano->etc->RouteManager->RouteManager and right-click on RouteManager which displays 6 methods.

1. listAllRoutes(): Lists all routes created for this route manager.
2. addRoute(routeGUID, lookupUrl, srcDest, tgtCF, tgtDest, isDurSub, durSubID, clientID, jmsSelector, isJMSReplyToAsTarget, enableDMQOnRoute, msgPropertyForTargetCF): Adds a route with given parameters.
3. addRoute(routeGUID, lookupUrl, srcDest, tgtCF, tgtDest, isDurSub, durSubID, clientID, jmsSelector, isJMSReplyToAsTarget, enableDMQOnRoute, msgPropertyForTargetCF, srcServerUserName, srcServerPassword, tarServerUserName, tarServerPassword): Adds a route with given parameters.
4. addRoute(routeGUID, lookupUrl, srcDest, tgtCF, tgtDest, isDurSub, durSubID, clientID, jmsSelector, isJMSReplyToAsTarget, enableDMQOnRoute, msgPropertyForTargetCF, dmqExpiryTime): Adds a route with given parameters.
5. removeRoute(routeGUID): Removes the route with specified routeGUID
6. saveRoutes(): This operation will save all the routes present in the server. Routes which are saved during earlier save operation will be overwritten. If server crashes during this operation, then output of this operation cannot be determined and during the next boot up of the server, routes may not be created properly. If target server is not running for any route during this server boot up then route creation fails."

36.3 Configuring Parameters Through WMT

Configure Routes contains information about the properties of the route manager. The table displays various properties of the route manager like MaxTopicBuffer, NumberOfRoutesExistingInServer, MaxCreateSessionTries, and so on.

If there are any routes already present on the server, they will be in the Route GUID list box.


FioranoMQ Web Manager - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://192.168.1.46:1780/#ConfigureRoutes
Google
User:admin
Fiorano
WEB MANAGEMENT CONSOLE
FioranoMQ 9.2.0, Build No. 9313, JVM Version : 32bit
Server Startup Time : Sat Mar 13 15:04:42 IST 2010
SaveConfigurations Shutdown Restart Logout
Users
Groups
Topics
Queues
ConnectionFactories
Policies
AuditManager
Routes
ConfigureRoutes
Select Action : < Select > Go

Configure Routes of FioranoMQ

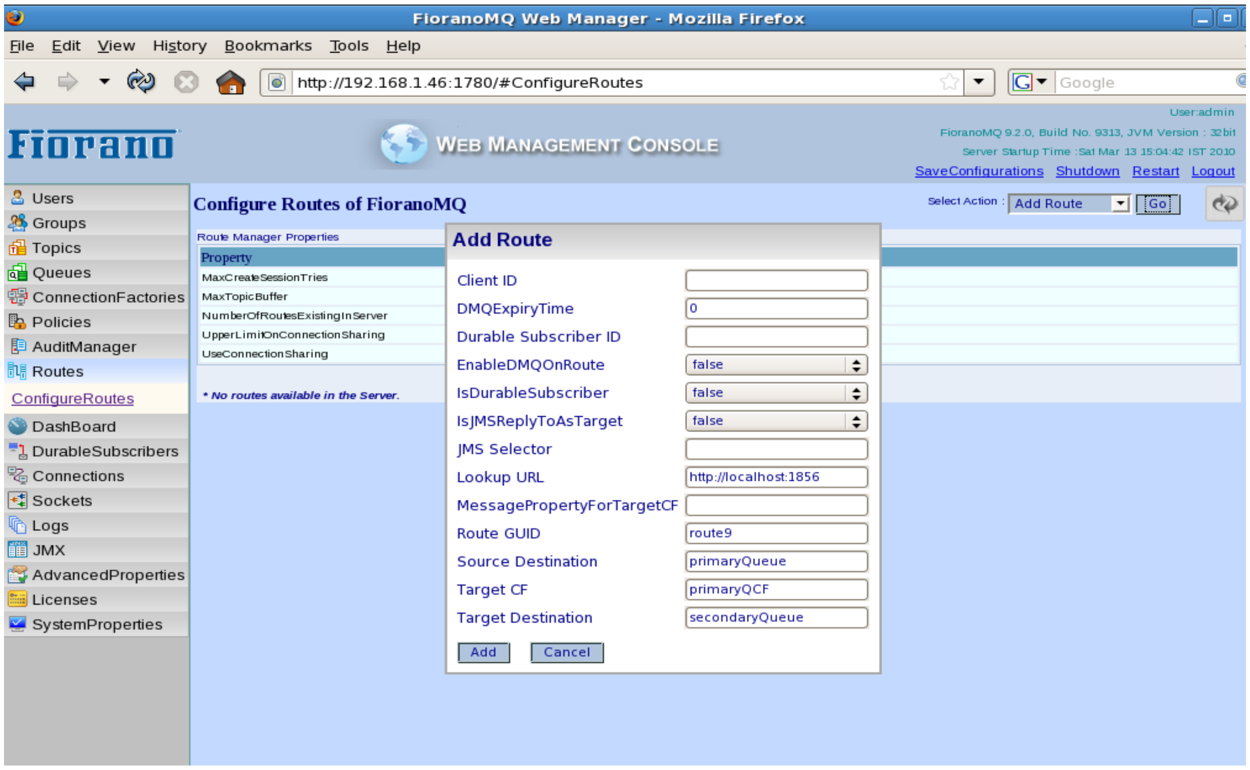
Route Manager Properties

Property	Value
MaxCreateSessionTries	1
MaxTopicBuffer	131072
NumberOfRoutesExistingInServer	0
UpperLimitOnConnectionSharing	10
UseConnectionSharing	true

** No routes available in the Server.*

Figure: Route

To Add or delete a route from the server, the appropriate option must be chosen from the Select Action drop-down menu.



FioranoMQ Web Manager - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://192.168.1.46:1780/#ConfigureRoutes

User:admin
FioranoMQ 9.2.0, Build No. 9313, JVM Version : 32bit
Server Startup Time :Sat Mar 13 15:04:42 IST 2010
SaveConfigurations Shutdown Restart Logout

Configure Routes of FioranoMQ

Select Action : Add Route Go

Route Manager Properties

Property
MaxCreateSessionTries
MaxTopicBuffer
NumberOfRoutesExistingInServer
UpperLimitOnConnectionSharing
UseConnectionSharing

* No routes available in the Server.

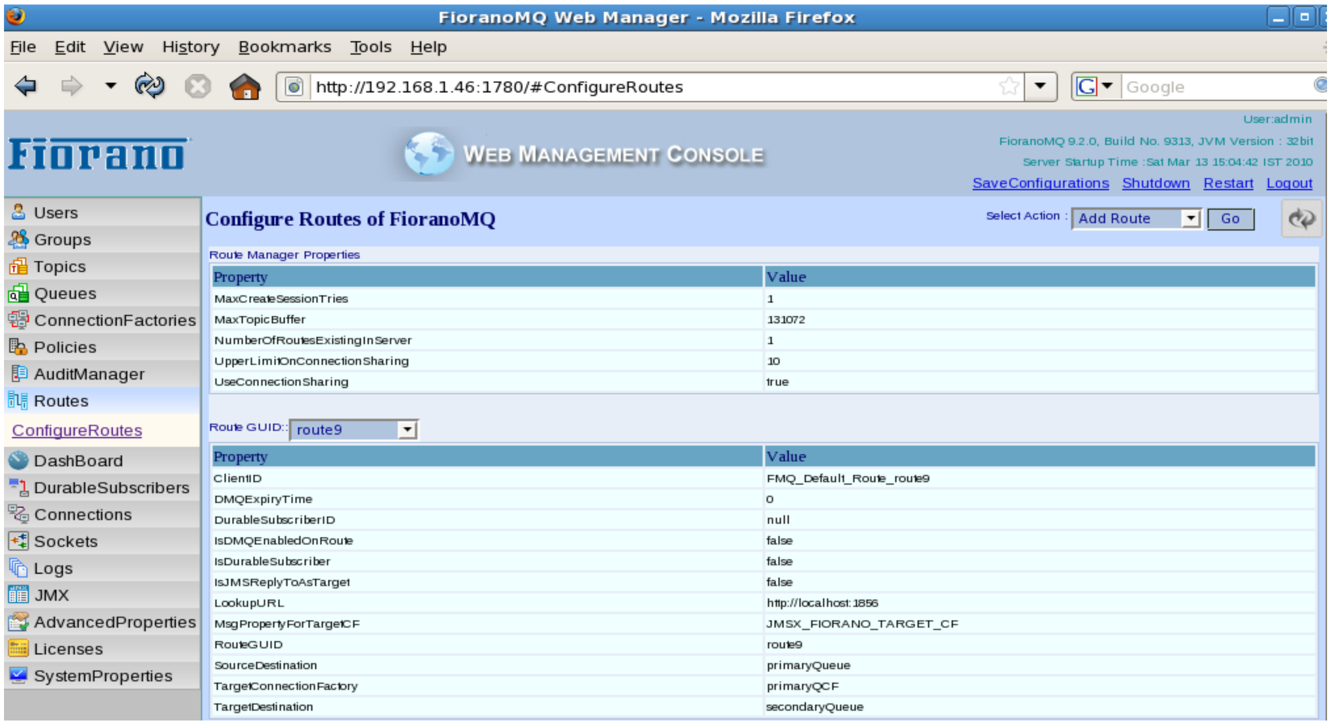
Add Route

Client ID	
DMQExpiryTime	0
Durable Subscriber ID	
EnableDMQOnRoute	false
IsDurableSubscriber	false
IsJMSReplyToAsTarget	false
JMS Selector	
Lookup URL	http://localhost:1856
MessagePropertyForTargetCF	
Route GUID	route9
Source Destination	primaryQueue
Target CF	primaryQCF
Target Destination	secondaryQueue

Add Cancel

Figure: Adding Route

On selecting any of the routes present in the Route GUID list box the properties of the corresponding route are displayed in the table below the list box.



FioranoMQ Web Manager - Mozilla Firefox

http://192.168.1.46:1780/#ConfigureRoutes

User:admin
FioranoMQ 9.2.0, Build No. 9313, JVM Version : 32bit
Server Startup Time :Sat Mar 13 15:04:42 IST 2010
SaveConfigurations Shutdown Restart Logout

Configure Routes of FioranoMQ

Select Action : Add Route Go

Route Manager Properties

Property	Value
MaxCreateSessionTries	1
MaxTopicBuffer	131072
NumberOfRoutesExistingInServer	1
UpperLimitOnConnectionSharing	10
UseConnectionSharing	true

Route GUID: route9

Property	Value
ClientID	FMQ_Default_Route_route9
DMQExpiryTime	0
DurableSubscriberID	null
IsDMQEnabledOnRoute	false
IsDurableSubscriber	false
IsJMSReplyToAsTarget	false
LookupURL	http://localhost:1856
MsgPropertyForTargetCF	JMSX_FIORANO_TARGET_CF
RouteGUID	route9
SourceDestination	primaryQueue
TargetConnectionFactory	primaryQCF
TargetDestination	secondaryQueue

36.4 Configuring Parameters Through XML file

The RouteUtility tool creates/removes multiple routes between destinations (Queues/Topics) based on the configuration file specified while running the route utility. The configuration file (routes.xml) contains the route properties of the each route to be created.

'routes.xml' file is located at $FIORANO_HOME/fmq/Utilities/RouteUtility/conf folder. If one of the route configuration is not specified with proper values, remaining routes can still be created/removed.

36.4.1 Command to Create/Remove Routes

UNIX:

```
$FIORANO_HOME/fmq/bin/routeUtility.sh [-operation <operation> -configFile
<routeConfigsFile>]
```

WINDOWS:

```
%FIORANO_HOME%/fmq/bin/routeUtility.bat [-operation <operation> -configFile
<routeConfigsFile>]
```

ConfigFile:

XML file which contains configuration of the routes. If no configuration file is specified while running the route Utility then 'routes.xml' will be taken as the default configuration file.

Operation: Operation that needs to be executed using the route utility.

Valid Values: CreateRoutes or removeRoutes.

If no operation is specified, then this will take 'createRoutes' as its operation.

36.4.2 Modifying routes.xml to Add/Delete Routes

‘route.xml’ provided with the RouteUtility has been preconfigured to create multiple routes with a unique Guid for each route. In order to add more routes to the XML file, please modify the XML file as shown below:

In order to add a new route copy the below tag and edit it and append to the routes.xml file. The complete syntax shown below will add the new route.

```
<ro:Route guid="sample_route1">

<ro:SourceServer>

          <ro:Address>localhost</ro:Address>

           <ro:Port>1858</ro:Port>

           <ro:Destination>primaryQueue</ro:Destination>
```

```
            <ro:UserName>admin</ro:UserName>

            <ro:Password>passwd</ro:Password>

</ro:SourceServer>

<ro:TargetServer>

            <ro:LookupURL>http://localhost:1856</ro:LookupURL>

            <ro:CFName>primaryCF</ro:CFName>

            <ro:Destination>primaryTopic</ro:Destination>

            <ro:ClientID>null</ro:ClientID>

</ro:TargetServer>

            <ro:IsDurableSubscriber>false</ro:IsDurableSubscriber>

            <ro:DurableSubscriber id="SOME_ID">true</ro:DurableSubscriber>

            <ro:MessageSelector></ro:MessageSelector>

            <ro:EnableDMQOnRoute>false</ro:EnableDMQOnRoute>

</ro:Route>
```

In order to delete a route remove the tag corresponding to the route to be deleted from the route.xml file

Chapter 37: JVM Arguments

37.1 Heap Memory Settings

- **Xmx**: Can be used to set maximum java heap size that can be used by the program. A low value can cause frequent **OutOfMemory** errors which might make the FioranoMQ server unreliable or low performance if the program's heap memory is reaching the maximum heap size as the GC will be run more frequently.

  You may see the amount of memory you use exceeds the amount specified for the Xmx parameter. While Xmx limits the java heap size, java will allocate memory for other things, including a stack for each thread. It is not unusual for the total memory consumption of the VM to exceed the value of –Xmx.

  Default value for fmq server is 256 MB

  Example: -Xmx256m sets maximum heap size to 256 MB
- **Xms**: Can be used to set initial java heap size to be used by the program. This is useful if the program consumes a large amount of heap memory right from the start. This avoids the JVM to be constantly increasing the heap resulting in better performance.

  Default value for fmq server is 128 MB

  Example: -Xms128m sets minimum heap size to 128 MB

  The value must be a multiple of, and greater than, 1024 bytes (1KB).

37.2 Stack Size settings

Xss: Can be used to set the java thread stack size. Each thread in the JVM is allocated a stack. The stack size limits the number of threads per JVM. If the stack size it too large, it will result in memory running out as each thread is allocated more memory than required. If the stack space is too small, there will eventually be an exception “Stack Overflow” error. If the stack space is too large, there will eventually be an exception similar to “Unable to create native thread”, if the server tries to create more threads.

37.3 Jconsole

Local monitoring with jconsole is useful for development and prototyping. Using jconsole locally is not recommended for production environments, because jconsole itself consumes significant system resources. Rather, use jconsole on a remote system to isolate it from the platform being monitored.

To enable monitoring and management from remote systems, set this system property when you start the JVM:

com.sun.management.jmxremote.port=portNum

Where, portNum is the port number through which you want to enable JMX/RMI connections. Be sure to specify an unused port number.

For more information on usage of jconsole refer to http://download.oracle.com/javase/1.5.0/docs/guide/management/jconsole.html

Note: The JVM hotspot settings mentioned below are non-standard hotspot options provided by SUN VM implementation and are not guaranteed to be supported by all JVM implementations e.g. IBM JRE does not support many of these options.

37.4 PermGen space

The permanent generation is used to hold reflective objects of the VM such as class objects and method objects. These reflective objects are allocated directly into the permanent generation, and it is sized independently from the other generations.

- **-XX:PermSize**: This option is used to set a new initial size on Sun JVM when starting the virtual machine.
- **-XX:MaxPermSize**: This option is used to set the maximum permanent generation size.

When the number of components launched in-memory increases, so do the number of classes loaded. Perm Gen space stores the meta data of the JVM and is not part of the Heap space.

37.5 -d64

Specifies whether the program is to be run in a 32-bit or 64-bit environment if available. If neither -d32 nor -d64 is specified, the default is to run in a 32-bit environment, except for 64-bit only systems.

37.6 Garbage Collection

- **-Xnoclassgc**: This is used to disable class garbage collection.

  This may lead to irrecoverable out of memory errors, and unless the reason for setting this flag is known thoroughly along with the consequences, setting this flag is not recommended.
- **-XX:MaxGCPauseMillis**: Sets the maximum desired number of milliseconds the garbage collector can pause the VM. This is only respected for the "throughput collector" or "parallel collector." In CMS, it has no effect. There is no default for this setting, so it has to be set for the garbage collector to meet a particular goal. However, it tries to meet this goal before meeting the GCTimeRatio goal.
- **-XX:GCTimeRatio**: Sets a goal for the throughput collector to avoid spending more than a certain amount of time doing garbage collection.

  By default the value is 99, meaning the application should get at least 99 times as much time as the collector. That is, the collector should run for not more than 1% of the total time.

- **-XX:-DisableExplicitGC**: Disable calls to System.gc(), JVM still performs garbage collection when necessary.

  It is dangerous for a misbehaving in-memory component to continuously issue System.GC calls when the Heap memory allocated is large. This option will disable actions against explicit calls. The memory of the server is best managed by the VM.

37.6.1 GC Algorithms

- **-XX:+UseConcMarkSweepGC**: This concurrent low pause collector is used to collect the tenured generation and does most of the collection concurrently with the execution of the application. The application is paused for short periods during the collection. The application is paused for short periods during the collection. A parallel version of the young generation copying collector is used with the concurrent collector.

  Works well for long running servers with a large heap memory that can afford to share CPU cycles with the garbage collector. Would result in the lowest pause times.
- **-XX:+CMSIncrementalMode**: This flag enables the incremental mode.

  Note: This works only when concurrent collector is enabled.

  For systems with 1 or 2 processors, this might make CMS run more smoothly and avoid concurrent mode failures because it actually does tiny stop-the-world phases to do some of its concurrent work instead of relying on the OS to give it an appropriate time-slice for a thread.
- **-XX:+UseParallelGC**: Selects the parallel garbage collector for the new generation of the Java heap . This collector is also referred to as the throughput collector. It uses a parallel version of the young generation collector. The old (tenured) generation is still cleaned with the default collector.

  By default on a host with N CPUs, the parallel garbage collector uses N garbage collector threads in the collection. The number of garbage collector threads can be controlled with a command line option -XX:ParallelGCThreads=<desired number>

  Note: -XX:+UseParallelGC should not be used with -XX:+UseConcMarkSweepGC .

37.6.2 GC logging

- -Xloggc:filename : Directs a smaller set of data to a file specified by filename. For every garbage collection, the following six fields are printed to the log file:
  - Cumulative time since data collection began (in seconds)
  - Garbage collection type
  - Heap in use before the GC event
  - Heap in use after the GC event
  - Current maximum heap size
  - Duration of the GC event

- **-XX:+PrintGCDetails**: More data is available if you start the JVM with the -XX:+PrintGCDetails or -XX:+PrintHeapAtGC options.

  Instructs the VM to be more verbose when printing out garbage collection data. Specifically it does not only tell you that there was a collection, but also what impact it had on the different generations. This flag is very useful when tuning generation sizes. In conjunction with -Xloggc this is the best setting for the free GCViewer.
- **-XX:+PrintGCDetails -Xloggc**: filename prints the details into file with name specified by filename
- **-XX:+PrintGCTimeStamps**: Prints the times at which the GC happens relative to the start of the application. Available only from J2SE1.4.0
- -**XX:+PrintTenuringDistribution**: This it used to generate the aging information of objects in young generation. It shows the threshold and the ages of objects in the new generation. It is also useful for observing the lifetime distribution of an application.

37.7 Heap Dump Settings

- **-XX:HeapDumpPath**: By default the heap dump is created in a file called java_pid<pid>.hprof in the working directory of the VM, where <pid> is the process ID. You can specify an alternative file name or directory with this option. This option is available only after JVM version 1.5 update 7 and above.

  For example, -XX:HeapDumpPath=~/dumps will cause the heap dump to be generated in the ~/dumps directory.
- -**XX:+HeapDumpOnOutOfMemoryError**: This VM option tells the VM to generate a heap dump when OutOfMemoryError is thrown because the java heap or the permanent generation is full. A heap dump is useful in production systems where you need to diagnose an unexpected failure.

37.8 Debugging

- -**XX:+PrintConcurrentLocks**: This will cause the Ctrl-Break handler to print a list of concurrent locks owned by each thread
- **-XX:+PrintClassHistogram**: This will cause the Ctrl-Break handler to print a heap histogram.
- **-XX:+TraceClassLoading**: This is used to trace loading of classes.
- **-XX:+TraceClassUnloading**: This is used to trace unloading of classes.

37.9 CoreDump on Application or JVM Error

-XX:ErrorFile=filename: If an error occurs, it saves the error data to the file specified where filename is the file location where error data need to be stored.

Note: Specific to Java 6.0, cannot be used with 1.5

Chapter 38: Admin Object's Configurations

38.1 Introduction

FioranoMQ installation comes up with a set of Destinations, Connections Factories, Users, and Groups with default configurations and Acls. One can either use the default set of Admin Objects by modifying their default configurations or create new Admin Objects with custom configurations.

New Admin Objects can be created in the following ways,

Using Fiorano tools like Studio and Web Console, but can be created one at a time manually.

Using JMX programs or JMS Admin connection, and can be programmed to created multiple Objects.

In order to reduce the programming efforts and possible coding errors while creating and configuring Admin Objects on a large scale, we need to have a functionality to read simple xml based configurations during the server startup.

Following Admin Objects and their properties can be configured through 'Admin Objects' xml file:

- Connection Factories and their metadata
- Destinations (Queues and Topics)and their properties
- Queue Subsystem properties
- Topic Subsystem properties
- Users and Groups
- Access Control Lists (Acls)

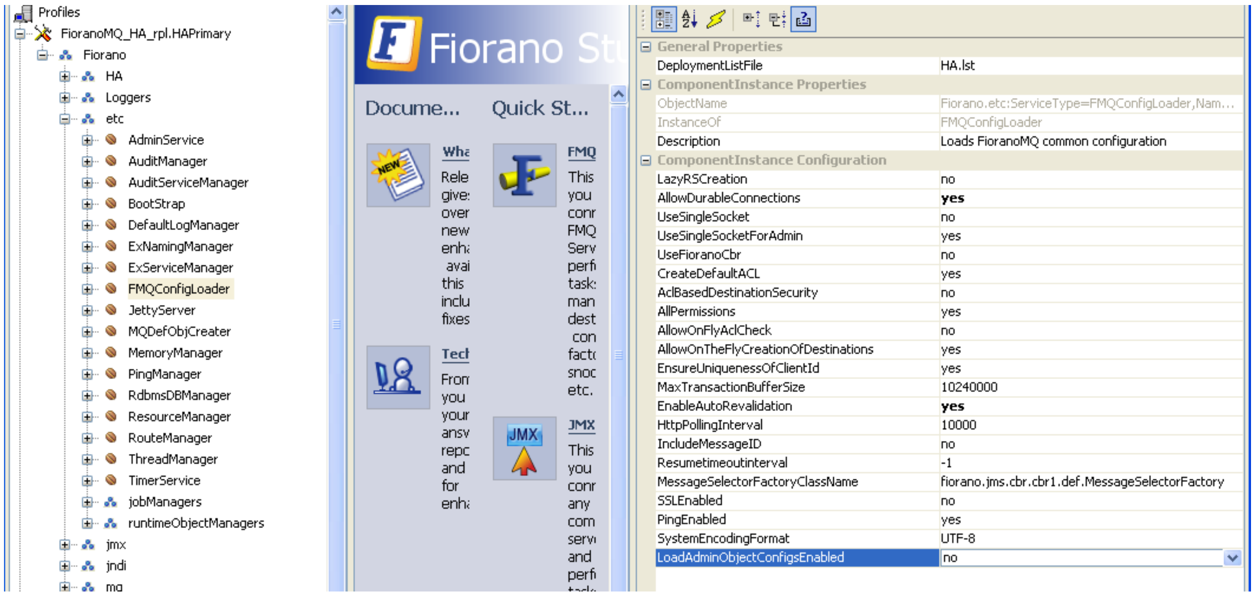
38.2 Enabling XML based configuration

Configuring Admin Objects through XML file by default is disabled and should be enabled explicitly through offline configurations. This section details on few other parameters related to this feature. These parameters can be configured through Fiorano Studio in offline mode.

LoadAdminObjectConfigsEnabled

This parameter can be located under the common configurations node 'FMQConfigLoader', which is responsible for enabling the feature to load Admin Objects and their configurations from an xml file. By default the parameter is disabled. Screen shot below shows the parameter location.

Default value: No



General Properties	
DeploymentListFile	HA.lst
ComponentInstance Properties	
ObjectName	Fiorano.etc:ServiceType=FMQConfigLoader,Nam…
InstanceOf	FMQConfigLoader
Description	Loads FioranoMQ common configuration
ComponentInstance Configuration	
LazyRSCreation	no
AllowDurableConnections	yes
UseSingleSocket	no
UseSingleSocketForAdmin	yes
UseFioranoCbr	no
CreateDefaultACL	yes
AclBasedDestinationSecurity	no
AllPermissions	yes
AllowOnFlyAclCheck	no
AllowOnTheFlyCreationOfDestinations	yes
EnsureUniquenessOfClientId	yes
MaxTransactionBufferSize	10240000
EnableAutoRevalidation	yes
HttpPollingInterval	10000
IncludeMessageID	no
Resumetimeoutinterval	-1
MessageSelectorFactoryClassName	fiorano.jms.cbr.cbr1.def.MessageSelectorFactory
SSLEnabled	no
PingEnabled	yes
SystemEncodingFormat	UTF-8
LoadAdminObjectConfigsEnabled	no

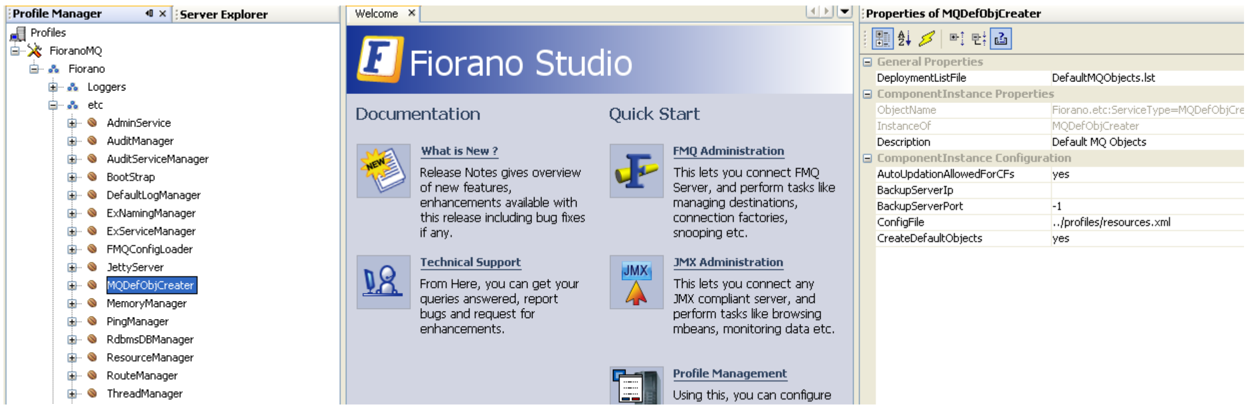
ConfigFile

The MBean ‘MQDefObjCreator’ is responsible for creating and configuring Admin objects using the xml file. The parameter ‘ConfigFile’ takes the xml file path with all the Admin objects to be created and configured. The xml file provided will be validated against the default XSD files located under the profiles home directory, $Fiorano_home/fmq/profiles/Schema.

Default value: ../profiles/resources.xml

CreateDefaultObjects

Disabling this parameter will not create the default Admin objects during the server startup. By default this value is set to ‘Yes’ to create all the default Admin Objects. The parameter can be located under the ‘MQDefObjCreator’ node, please see the screen shot below for your reference.



Properties of MQDefObjCreater

General Properties	
DeploymentListFile	DefaultMQObjects.lst
ComponentInstance Properties	
ObjectName	Fiorano.etc:ServiceType=MQDefObjCre
InstanceOf	MQDefObjCreater
Description	Default MQ Objects
ComponentInstance Configuration	
AutoUpdationAllowedForCFs	yes
BackupServerIp	
BackupServerPort	-1
ConfigFile	../profiles/resources.xml
CreateDefaultObjects	yes

Configuration Log

All the logs related to configurations are redirected to “MQAdminObjects.out” under the default logs directory. The configuration log can be used to verify the XML validation failures if any, Admin objects configurations details and errors that encountered during the creation or configuration of Admin Objects.

JMX Operations

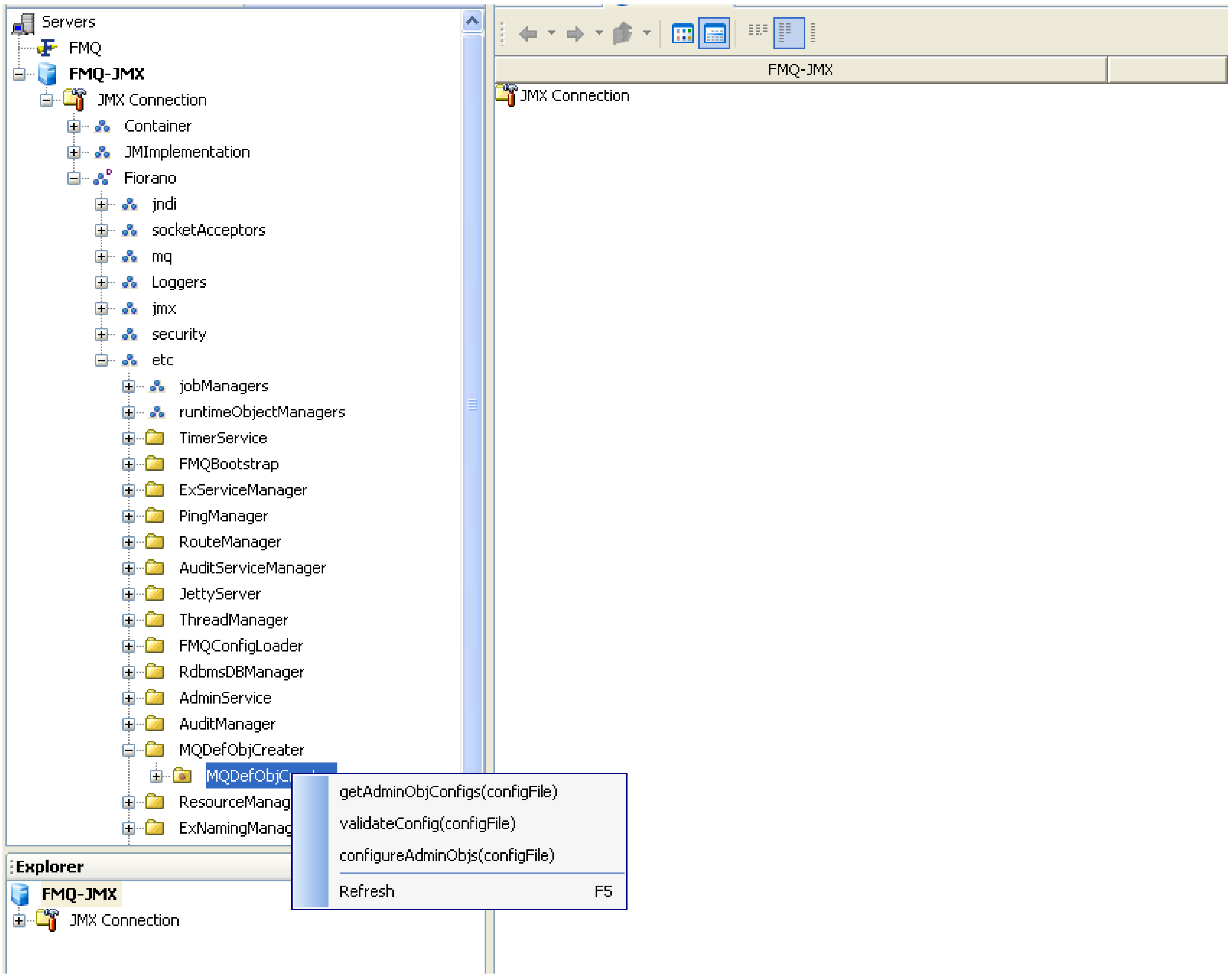
JMX operations have been provided on ‘MQDefObjCreator’ MBean to validate Admin Objects XML file, to extract the existing Admin Objects configurations in the form of an Xml file, and also provision to load the configurations through JMX api in addition to the way they are loaded during the server startup.

- getAdminObjConfigs (String ConfigFile): Loads the modified or newly created admin objects (Destinations, Connection Factories, Security) and their configurations as per the schema under the active server's profile directory.
- ConfigFile: The xml filename to which the configurations are persisted. The file is by default saved to $Fiorano_Home/fmq/profiles/<profile_name> directory.
- validateConfig (String configFilePath): The API can be used for validating the given XML file against the Schema files located at $Fiorano_Home/fmq/profiles/Schema directory.
- ConfigFilePath: Path of the xml configuration file to validate.

  Returns ‘true’ if the file is successfully validated and ‘false’ if it is unsuccessful. The corresponding errors logs when a validation fails can be found in ‘MQAdminObjects.out’ log file.
- configureAdminObjs(String configFile): The API can be used for configuring Admin objects by providing the path for xml file. The xml file is configured after validating it against the schema.
- configFile: Path of the xml configuration file to validate and configure.

  Returns ‘true’ if it is configured successfully and returns ‘false’ if there is any error in the process of validation or configuration. All the relevant errors or exceptions can be found in ‘MQAdminObjects.out’ log file.

The following figure displays the JMX operations:



Note: Please refer to the FioranoMQ reference guide for all the valid value to be provided for each parameter while configuring Connection factories, Destinations, and Subsystem level parameters.

Schema

Following are the list of schema files used for validating the xml file for configuring the Admin Objects during the server startup or through the above mentioned JMX APIs.

AdminObjects.xsd

The ‘AdminObjects’ node is the root element for all other nodes such as ‘ConnectionFactory’, ‘Destination’, ‘QueueSubsystem’ , ‘TopicSubsystem’, and ‘SecurityRealm’.

CFParameters.xsd

The schema file lists the CF parameters, which are imported to ‘AdminObjects.xsd’.

QueueParams.xsd

This schema file lists the Queue parameters, and the corresponding data types accepted.

QueueSubSystemParams.xsd

This schema file supports the Queue subsystem parameters and the corresponding data types accepted.

TopicParams.xsd

This schema file lists the Topic parameters, and the corresponding data types accepted.

TopicSubSystemParams.xsd

This schema file supports the Topic subsystem parameters and the corresponding data types accepted.

38.3 Limitations with Admin Objects feature

1. The Connection Factory's metadata is only allowed to update at the time of creating the Connection Factory itself through Admin Objects feature, which includes, Type, Description, and Auto Update.
2. However, in order to re-update the 'AutoUpdationAllowed' parameter, please use the following entries in CFParameters.xsd and factory.xml file respectively.

   <xs:element name="AutoUpdationAllowed" type="xs:boolean"/>
   and
   <tns:AutoUpdationAllowed>false</tns:AutoUpdationAllowed>
3. Creating XA Connection Factories is not supported with the current version of FioranoMQ.
4. Following corrections or workarounds (in xsd and xml files respectively) needs to be done for rest of the parameters, which will be corrected in the next version of FioranoMQ.

38.3.1 Connection Factory Parameters

Please note that the first line is the change to be done in Schema file (.xsd) and the second line is the corresponding xml file entry.

ConnectURL

```
<xs:element name="ConnectURL" type="xs:string" />
<tns:ConnectURL>http://localhost:1856</tns:ConnectURL>
```

ShutdownHookEnabled

```
<xs:element name="ShutdownHookEnabled" type="xs:boolean" default="true" />
<tns:ShutdownHookEnabled>true</tns:ShutdownHookEnabled>
```

BatchTimeoutInterval

```
<xs:element name="BatchTimeoutInterval" type="xs:string" />  (changed from time
datatype)
```

CreateLocalSocket

```
<xs:element name="CreateLocalSocket" type="xs:string" /> (changed from Boolean)
```

DurableConnectionsBaseDir

```
<xs:element name="DurableConnectionsBaseDir" type="xs:string" />
<tns:DurableConnectionsBaseDir>../csp</tns:DurableConnectionsBaseDir>
```

EnableLMS

```
<tns:EnableLMS>false</tns:EnableLMS>
<xs:element name="EnableLMS" type="xs:boolean" default="true" />
```

PublishWaitDuringSendPendingCSP

```
<xs:element name="PublishWaitDuringSendPendingCSP" type="xs:string" />
<tns:PublishWaitDuringSendPendingCSP>5</tns:PublishWaitDuringSendPendingCSP>
```

useThreadContextClassLoader

```
<xs:element name="useThreadContextClassLoader" type="xs:boolean" default="false" />
<tns: useThreadContextClassLoader >true</tns: useThreadContextClassLoader >
```

UseFioranoCbr

```
<xs:element name="UseFioranoCbr" type="xs:boolean" default="false" />
<tns:UseFioranoCbr>true</tns:UseFioranoCbr>
```

AsyncSendBatchBufferSize

```
<xs:element name="AsyncSendBatchBufferSize" type="xs:string" />
<tns:AsyncSendBatchBufferSize>1000</tns:AsyncSendBatchBufferSize>
```

AsyncSendCompletionWaitTimeout

```
<xs:element name="AsyncSendCompletionWaitTimeout" type="xs:string" />
<tns:AsyncSendCompletionWaitTimeout>10</tns:AsyncSendCompletionWaitTimeout>
```

PublisherBehaviourOnAsyncSendBufferOverflow

```
<xs:element name="PublisherBehaviourOnAsyncSendBufferOverflow" type="xs:string" />
<tns:PublisherBehaviourOnAsyncSendBufferOverflow>block</tns:PublisherBehaviourOnAsyncS
endBufferOverflow>
```

38.3.2 Queue Parameters

Please note that the first line is the change to be done in the Schema file (.xsd) and the second line is the corresponding xml file entry.

UseReAssignableMessageGroupID

```
<xs:element name="UseReAssignableMessageGroupID" type="xs:boolean" default="true" />
<tnsB:UseReAssignableMessageGroupID>false</tnsB:UseReAssignableMessageGroupID>
```

QueueIdleStateNotifierThresholdTime

```
<xs:element name="QueueIdleStateNotifierThresholdTime" type="xs:int" default="0"/>
<tnsB:QueueIdleStateNotifierThresholdTime>34</tnsB:QueueIdleStateNotifierThresholdTime
>
```

38.3.3 Queue Subsystem Parameters

MaxSessionCount

```
<xs:element name="MaxSessionCount" type="xs:int" default="-1" />
<QSubSys:MaxSessionCount>4</QSubSys:MaxSessionCount>
```

MaxSenderCount

```
<xs:element name="MaxSenderCount" type="xs:int" default="-1" />
<QSubSys:MaxSenderCount>4</QSubSys:MaxSenderCount>
```

MaxReceiverCount

```
<xs:element name="MaxReceiverCount" type="xs:int" default="-1" />
<QSubSys:MaxReceiverCount>3</QSubSys:MaxReceiverCount>
```

MaxTempQueueCount

```
<xs:element name="MaxTempQueueCount" type="xs:int" default="-1" />
<QSubSys:MaxTempQueueCount>4</QSubSys:MaxTempQueueCount>
```

QueueIdleStateNotifierThresholdTime

```
<xs:element name="QueueIdleStateNotifierThresholdTime" type="xs:int" />
<QSubSys:QueueIdleStateNotifierThresholdTime>30</QSubSys:QueueIdleStateNotifierThresho
ldTime>
```

Chapter 39: Frequently Asked Questions

This chapter contains compilations of information which are usually the result of certain questions constantly being asked. If you do not see your question here, feel free to post it to customer support.

Question: Which jar file is use to run client applications in distributed environment apart from jms.jar?

Answer: You can use the aggregated jars **fmq\lib\client\all\fmq-client.jar** and **framework\lib\all\fiorano-framework.jar** to run client applications in distributed environment.

Question: How to set log levels in a client application?

Answer: Enable Tracing in Clients, launch the client by setting the system property for any module for which the user wants to enable tracing.

For Example: DCSP = 5, when this system property is enabled in the client side then user can view trace messages in the client itself.

Question: How to set default property values for dynamically created Queue?

Answer: By default, all the parameters present in Queue are also present in QueuingSubSystem. In QueuingSubSystem for all properties present in Queue default values are -1 for numerical values and default value is 'DEFAULT' for boolean represented values. If these values are default values then whatever the values user set in Queue configuration property sheet will be taken. Otherwise that particular property value from QueuingSubSystem will be taken into account even though that value is modified locally for a particular queue. If the value of a particular parameter is modified to some other non-default value in QueuingSubSystem then that value will be written for all other existing queues in the server. If that value is modified again in QueuingSubSystem then the last value written for that queue will be considered. For example the value of 'InMemoryBufferSize' in all Queues by default is '1048576', if 'InMemoryBufferSize' is modified to 1024 in Queueingsubsystem, then that value will be written to all Queues. If the user modified this value to -1(default value) again then 1024 is taken for all queues and not the 1048576(default value). If the user wants other than 1024 for a particular queue then he can modify it for a particular queue. Modifying parameters at QueuingSubSystem level will be useful only if most of the Queues contain same values for most of the properties other than default value.

Note: Most of the parameters require server restart and Studio will warn the user if he is changing any parameter which requires server restart.

If the value of a particular parameter is modified to some other non-default value in QueuingSubSystem then that value will be written for all other existing queues in the server. If that value is modified again in QueuingSubSystem to default value then the last value written for that queue is considered.

Question: I cannot find any indication that the optional support for JMS priority ordering of messages is provided?

Answer: FioranoMQ has full support for priority based message queueing. The server ensures the sequence of delivery strictly based on the priority of the message i.e., all messages with higher priority get delivered first.

Question: How messages are distributed when there are multiple QueueReceivers for the same Queue?

Answer: The default behaviour is **roundrobin**, but it can vary depending on your prefetching configuration and session level throttling during high traffic.

Question: I offer a flow control mechanism, but it is only available for topics (pub/sub) not queues, why?

Answer: Several flow control mechanisms exist for PTP. The server is capable of throttling the sender when receive rate is slower than the send rate on the queue. The extent of such throttle on the sender is configurable. Server behavior in case it runs out of buffer space for non persistent messages is also configurable. When message prefetching is in use, the prefetch threshold and prefetch count is also configurable to maximize throughput.

Question: List the vender specific error codes which can be populated as Errorcode field of JMSException, thrown to a JMS Client.?

Answer: The following list contains the entire vendor specific JMSExceptions:

1. **VERSION_INCOMPATIBILITY_ERROR:** There has been a version mismatch between the Kernel and your Client application
2. **INVALID_ARGUMENTS_ERROR:** Invalid arguments passed to the method
3. **INVALID_ACKNOWLEDGMENT_MODE_SPECIFIED:** Invalid Acknowledgment Mode specified.
4. **INVALID_ADMINISTERED_OBJECT:** Error encountered while accessing the administered object
5. **INVALID_TIMEOUT_SPECIFIED_FOR_MONITOR:** timeout specified for a monitor can not be less than 10 sec
6. **UNABLE_TO_CREATE_USER_OR_GROUP_WITH_EMPTY_NAME:** Unable to create user/group with name as empty string
7. **INVALID_CONFIGURATION_OPTION_USED:** Invalid Configuration Option used
8. **INVALID_DESTINATION_HANDLE:** Destination Handle is invalid
9. **INVALID_LOOKUP_ERROR:** Invalid type specified for lookup
10. **INVALID_SUBSCRIPTION_NAME:** Invalid Subscription name
11. **NO_PRINCIPAL_SPECIFIED:** ACL Owner is not specified.
12. **LM_INVALID_TARGET_FILE:** Target file specified in the message is not valid
13. **LM_INVALID_SOURCE_FILE:** Source file specified in the message is not valid
14. **CLASS_NOT_FOUND_EXCEPTION:** Particular class not found
15. **NAME_OF_THE_CLASS_PASSED_IS_INVALID:** Name of the class passed is Invalid

16. **SECURITY_MANAGER_ERROR:** Error loading installed Security Manager
17. **INVALID_URL_FOR_CLIENT_PROXY_SPECIFIED:** Invalid URL for Client Proxy Specified
18. **INVALID_URL_FOR_SERVER_PROXY_SPECIFIED:** Invalid URL for Server Proxy Specified
19. **PROVIDER_URL_PARSING_ERROR:** Error encountered while trying to parse the Provider URL.
20. **SECURITY_EXCEPTION_ERROR:** Security Exception occurred.
21. **ERROR_PUBLISHING_MESSAGE:** Message could not be published.
22. **CREATE_DURABLE_SUBSCRIBER_FAILURE:** Can not create a durable subscriber on a temporary topic
23. **OPERATION_UNSUPPORTED_ERROR:** This operation is not supported presently.
24. **LM_CSP_ENABLED:** Large message transfer not allowed when CSP is enabled
25. **UNBOUND_CONTEXT_ERROR:** InitialContext is null:: Not bound
26. **NOT_OWNER_ERROR:** Specified principal is not an owner of the ACL
27. **ACKNOWLEDGE_FAILURE_ERROR:** Failed to acknowledge the message,some exception occurred.
28. **BIND_FAILURE_ERROR:** Error encountered while trying to execute bind call.
29. **CLASS_NOT_INITIALIZED_EXCEPTION :** lass has not been initialized
30. **DCONNECTION_CONSUMER_COMMIT:** Unable to commit messages for durable connection consumer
31. **CONNECTION_CREATION_FAILURE_ERROR:** Error encountered while trying to create a new connection.
32. **CONNECTION_DESTROY_FAILURE_ERROR:** Error encountered while trying to destroy the connection abnormally.
33. **ERROR_COMMITTING_SESSION:** Session could not be committed.
34. **FAILED_TO_WAIT_ON_OBJECT:** Error encountered while waiting on an object to be notified.
35. **LEAST_LOADED_SERVER_CALCULATION_ERROR:** Failed to calculate the least loaded server.
36. **CONSUMER_STARTUP_FAILURE_ERROR:** Error encountered while trying to start the consumer.
37. **DCONNECTION_CONSUMER_ROLLBACK:** Unable to rollback messages for durable connection consumer
38. **DCONNECTION_CONSUMER_START:** Unable to start durable durable connection consumer
39. **OBJECT_WRITE_FAILURE_ERROR:** Error encountered while trying to write the object
40. **UNABLE_TO_LOAD_CLASS:** Unable to find [or create instance of] class
41. **SECURITY_MANAGER_ERROR:** Error loading installed Security Manager
42. **EXECUTE_JOB_ERROR:** Error encountered while trying to execute SendJob.

43. **MESSAGE_PROPERTY_SET_ERROR:** Unable to set a header or property field for the message.
44. **LM_UNABLE_TO_RESUME_RECEIVE:** Exception occurred while resuming receive
45. **LM_TRANSFER_CANCELLED:** Transfer forcefully canceled by the user.
46. **LM_TRANSFER_STOPPED:** Transfer forcefully stopped by the user.
47. **PINGER_NOT_STARTED:** Pinger is not started.
48. **LM_READ_DATA_ERROR:** Exception occurred while reading data from source file
49. **LM_WRITE_DATA_ERROR:** Exception occured while writing data on the target file
50. **SERVER_HANDSHAKE_ERROR** : Problem in handshaking with the server
51. **LM_UNABLE_TO_SAVE_TARGET_FILE:** Unable to save target file. Probably, there is not enough disk space available.
52. **ERROR_KRPC_INITIALIZATION:** Problem in initializing krpc
53. **LOCALHOST_CONFIGURATION_LOADING_ERROR:** Error encountered while trying to load localhost configuration.
54. **SERVICE_ID_WRITE_FAILURE:** Failed in writing the Service ID to Kernel
55. **MESSAGE_READ_ERROR :** Error encountered while trying to read the message
56. **SET_NAGLES_BIT_FAILURE_ERROR:** Error encountered while trying to set the Nagles bit.
57. **CONSUMER_STOP_TIMEOUT:** Unable to stop the consumer's delivery of messages to listener within specified timeout.
58. **SESSION_STOP_TIMEOUT:** Unable to stop the session's delivery of messages to listener within specified timeout.
59. **LM_REQUEST_TIMEOUT:** No request received from any consumer in the given time
60. **USER_THREAD_INTERRUPT_ERROR:** User Thread,doing the operation,interrupted
61. **KERNEL_IO_ERROR:** Kernel I/O error occurred
62. **ERROR_PUBLISHING_MESSAGE_GROUP:** Unable to send message group to the server.
63. **PUBLISH_FAILURE_RECONNECT_ACTIVE:** Failed to send the messages as connection with the server is down. However client is attempting to reconnect to the server
64. NULL_SOCKET_ENCOUNTERED: Socket is null.

Question: When running multiple Publisher/Subscriber applications, while closing Subcribers we get the following exception:

"Unable to stop the consumer's delivery of messages to listener within specified timeout. :: Message delivery took longer then 30,000."

Answer: This is harm less exception. This usually occurs when subscriber.close() is called and subscriber is not able to deliver the message within 30 seconds to the message listeners. Further there will not be any message loss because of this exception.

Question: How to enable the FioranoMQ web console (on port 1780).

Answer: To enable the FioranoMQ Webconsole on port 1780, perform the following steps:

1. For enabling the MQ Web console, the jetty (HTTP) server needs to be started on the peer profile.
2. Open Profile. Navigate to {PROFILE_NAME}->Fiorano->etc->JettyServer. Set EnableStart to true and set portnumber if the default need to be changed.
3. If Standalone Peer Profiles are being used, then the Enterprise Server's Peer Repository needs to be cleaned up before the Peer is started up. If Peer is run in HA Mode, then this step can be skipped. HA Profile is configured to always push the configuration with the FES and standalone servers pull the profile and replace its own from the FES repository.
4. When the peer starts up you should see a line stating that webconsole is available in 1780 port.

Question: I see a lot in the profiles about RDBMS usage for the MQ. I don't want to draw the wrong conclusion from it, but should I take the PTP/PUBSUB storage in an RDBMS is supported?

Answer: MQ PTP/Pub sub Persistence store can be configured as RDBMS Store as well. The only caution I would like to throw in here is the Message Delivery performance of the server will be affected by choosing a RDBMS based store over the default file based store which is tailor made for high performance.

Question: Once a new Dispatcher member in the server is created, the Dispatcher member server MBeans is not visible in the Fiorano Studio or JMX View of web console?

Answer: The MBeans related to the configuration objects for the newly added dispatcher member servers are only created at the time of start-up of the server.

Therefore, they will not be listed in the JMX View of Studio/WebConsole until the server is restarted. These configuration MBeans include those for Dispatcher (new servers), Repeater (new links or new topic configurations) and Bridge (new links or new channels).